Lecture Notes in Computer Science 3824

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison
Lancaster University, UK

Takeo Kanade
Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler
University of Surrey, Guildford, UK

Jon M. Kleinberg
Cornell University, Ithaca, NY, USA

Friedemann Mattern
ETH Zurich, Switzerland

John C. Mitchell
Stanford University, CA, USA

Moni Naor
Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz
University of Bern, Switzerland

C. Pandu Rangan
Indian Institute of Technology, Madras, India

Bernhard Steffen
University of Dortmund, Germany

Madhu Sudan
Massachusetts Institute of Technology, MA, USA

Demetri Terzopoulos
New York University, NY, USA

Doug Tygar
University of California, Berkeley, CA, USA

Moshe Y. Vardi
Rice University, Houston, TX, USA

Gerhard Weikum
Max-Planck Institute of Computer Science, Saarbruecken, Germany

Laurence T. Yang Makoto Amamiya
Zhen Liu Minyi Guo
Franz J. Rammig (Eds.)

Embedded and Ubiquitous Computing – EUC 2005

International Conference EUC 2005
Nagasaki, Japan, December 6-9, 2005
Proceedings

Springer

Volume Editors

Laurence T. Yang
St. Francis Xavier University, Department of Computer Science
Antigonish, NS, B2G 2W5, Canada
E-mail: lyang@stfx.ca

Makoto Amamiya
Kyushu University, Faculty of Information Science and Electrical Engineering
Department of Intelligent Systems
6-1 Kasuga-Koen, Kasuga, Fukuoka 816-8580, Japan
E-mail: amamiya@al.is.kyushu-u.ac.jp

Zhen Liu
Nagasaki Instiute of Applied Science, Graduate School of Engineering
536 aba-machi, Nagasaki 851-0193, Japan
E-mail: liuzhen@cc.nias.ac.jp

Minyi Guo
University of Aizu, Department of Computer Software
Aizu-Wakamatsu City, Fukushima 965-8580, Japan
E-mail: minyi@u-aizu.ac.jp

Franz J. Rammig
University of Paderborn, Heinz Nixdorf Institute
33102 Paderborn, Germany
E-mail: franz@uni-paderborn.de

Library of Congress Control Number: 2005936806

CR Subject Classification (1998): C.2, C.3, D.4, D.2, H.4, H.3, H.5, K.4

ISSN 0302-9743
ISBN-10 3-540-30807-5 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-30807-2 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

springeronline.com

© IFIP International Federation for Information Processing 2005
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11596356 06/3142 5 4 3 2 1 0

Preface

Welcome to the proceedings of the 2005 IFIP International Conference on Embedded and Ubiquitous Computing (EUC 2005), which was held in Nagasaki, Japan, December 6–9, 2005.

Embedded and ubiquitous computing is emerging rapidly as an exciting new paradigm to provide computing and communication services all the time, everywhere. Its systems are now pervading every aspect of life to the point that they are hidden inside various appliances or can be worn unobtrusively as part of clothing and jewelry. This emergence is a natural outcome of research and technological advances in embedded systems, pervasive computing and communications, wireless networks, mobile computing, distributed computing and agent technologies, etc. Its tremendous impact on academics, industry, government, and daily life can be compared to that of electric motors over the past century, in fact it but promises to revolutionize life much more profoundly than elevators, electric motors or even personal computers.

The EUC 2005 conference provided a forum for engineers and scientists in academia, industry, and government to address profound issues including technical challenges, safety, and social, legal, political, and economic issues, and to present and discuss their ideas, results, work in progress, and experience on all aspects of embedded and ubiquitous computing.

There was a very large number of paper submissions (376), not only from Europe, but also from Asia and the Pacific, and North and South America. All submissions were reviewed by at least three Program or Technical Committee members or external reviewers. It was extremely difficult to select the presentations for the conference because there were so many excellent and interesting submissions. In order to allocate as many papers as possible and keep the high quality of the conference, we finally decided to accept 114 papers for oral presentations. We believe that all of these papers and topics not only provided novel ideas, new results, work in progress, and state-of-the-art techniques in this field, but also stimulated future research activities in the area of embedded and ubiquitous computing.

The exciting program for this conference was the result of the hard and excellent work of many others, such as Program Vice-chairs, external reviewers, Program and Technical Committee members, all working under a very tight schedule. We were also grateful to the members of the Organizing Committee for supporting us in handling many organizational tasks, and to the keynote speakers for accepting to come to the conference with enthusiasm. Last but not least, we hope you enjoy the conference proceedings.

October 2005

Laurence T. Yang, Mokoto Amamiya
Zhen Liu, Minyi Guo and Franz J. Rammig
EUC 2005 Program and General Chairs

Organization

EUC 2005 was organized and sponsored by the Nagasaki Institute of Applied Science (NiAS), Japan and International Federation for Information Processing (IFIP). It was held in cooperation with the IEEE Computer Society, IEICE Information and System Society, *Lecture Notes in Computer Science* (LNCS) of Springer, and The Telecommunications Advancement Foundation (TAF).

Executive Committee

General Chairs:	Zhen Liu, Nagasaki Institute of Applied Science, Japan
	Franz J. Rammig, University of Paderborn, Germany
Program Chairs:	Laurence T. Yang, St. Francis Xavier University, Canada
	Mokoto Amamiya, Kyushu University, Japan
Program Vice-chairs:	Vipin Chaudhary, Wayne State University, USA
	Jingling Xue, University of New South Wales, Australia
	Giorgio Buttazzo, University of Pavia, Italy
	Alberto Macii, Politecnico di Torino, Italy
	Xiaohong Jiang, Tohoku University, Japan
	Patrick Girard, LIRMM, France
	Lorenzo Verdoscia, ICAR, National Research Council, Italy
	Jiannong Cao, Hong Kong Polytechnic University, China
	Ivan Stojmenovic, Ottawa University, Canada
	Tsung-Chuan Huang, National Sun Yet-sen University, Taiwan
	Chih-Yung Chang, Tamkang University, Taiwan
	Leonard Barolli, Fukuoka Institute of Technology, Japan
	Hai Jin, Huazhong University of Science and Technology, China
Steering Committee:	Sajal K. Das, University of Texas at Arlington, USA
	Guang R. Gao, University of Delaware, USA
	Minyi Guo (Chair), University of Aizu, Japan
	Dayou Liu, Jilin University, China
	Zhen Liu, Nagasaki Institute of Applied Science, Japan
	Jinpeng Huai, Beihang University, China
	Jianhua Ma, Hosei University, Japan

Executive Committee (continued)

	Ryuzo Takiyama, Nagasaki Institute of Applied Science, Japan
	Xiaopeng Wei, Dalian University, China
	Laurence T. Yang (Chair), St. Francis Xavier University, Canada
Panel Chairs:	Jianhua Ma, Hosei University, Japan
	Pao-Ann Hsiung, National Chung Cheng University, Taiwan
Workshop Chairs:	Makoto Takizawa, Tokyo Denki University, Japan
	Seongsoo Hong, Seoul National University, Korea
Industrial Liaison:	Shih-Wei Liao, Intel, USA
	Zhaohui Wu, Zhejiang University, China
Publicity Chairs:	Hui Wang, University of Aizu, Japan
	Andrea Acquaviva, University of Urbino, Italy
Demo and Exhibition:	Tomoya Enokido, Rissho University, Japan
Tutorial Chairs:	Beniamino Di Martino, Second University of Naples, Italy
	Chung-Ta King, National TsingHua University, Taiwan
Web Masters:	Shinichi Kamohara, Nagasaki Institute of Applied Science, Japan
	Noriyuki Kitashima, Nagasaki Institute of Applied Science, Japan
Publication Committee:	Haibo Yu (Chair), Kyushu University, Japan
	Tony Li Xu, St. Francis Xavier University, Canada
Local Organizing Chairs:	Kenichi Ito, Siebold University of Nagasaki, Japan
	Brian Burke-Gaffney, Nagasaki Institute of Applied Science, Japan
NIAS Executive	Ryuzo Takiyama (Chair), Susumu Yoshimura (Vice-chair)
Committee:	Yoshito Tanaka, Junichi Ikematsu, Brian Burke-Gaffey
	Makoto Shimojima, Noriyuki Kitajima, Teruyuki Kaneko
	Shinichi Kamohara, Takahiro Fusayasu, Kouji Kiyoyama
	Shinichiro Hirasawa, Saori Matsuo

Program/Technical Committee

Ben A. Abderazek	University of Electro-Communications, Japan
Jose Albaladejo	Polytechnical University of Valencia, Spain

Program/Technical Committee (continued)

Luis Almeida	University of Aveiro, Portugal
Giuseppe Anastasi	University of Pisa, Italy
Aldo Baccigalupi	University of Naples "Federico II", Italy
Juergen Becker	University of Karlsruhe, Germany
Davide Bertozzi	Università di Ferrara, Italy
Enrico Bini	Scuola Superiore Sant'Anna, Italy
Rajkumar Buyya	Melbourne University, Australia
Jean Carle	University of Lille, France
Sun Chan	ASTRI, Hong Kong, China
Chih-Yung Chang	Tamkang University, Taiwan
Naehyuck Chang	Seoul National University, Korea
Han-Chieh Chao	National Dong Hwa University, Taiwan
Jiann-Liang Chen	National Dong Hwa University, Taiwan
Yuh-Shyan Chen	National Chung Cheng University, Taiwan
Tzung-Shi Chen	National University of Tainan, Taiwan
Guihai Chen	Nanjing University, China
Jorge Juan Chico	Universidad de Sevilla, Spain
Li-Der Chou	National Central University, Taiwan
Sajal K. Das	University of Texas at Arlington, USA
Alex Dean	North Carolina State University, USA
Lawrence Y. Deng	St. John's and Mary's Institute of Technology, Taiwan
Giuseppe De Marco	Fukuoka Institute of Technology, Japan
Bjorn De Sutter	Ghent University, Belgium
Carlos Dominguez	Polytechnical University of Valencia, Spain
Chi-Ren Dow	Feng Chia University, Taiwan
Arjan Durresi	Lousiana State University, USA
Paal E. Engelstad	University of Oslo, Norway
Tomoya Enokido	Rissho University, Japan
Raffaele C. Esposito	University of Sannio, Italy
Jih-Ming Fu	Cheng-Hsiu University of Technology, Taiwan
Marisol Garcia Valls	Universidad Carlos III de Madrid, Spain
Luis J. Garcia Villalba	Complutense University of Madrid, Spain
Rung-Hung Gau	National Sun Yat-sen University, Taiwan
Antonio Gentile	University of Palermo, Italy
Luis Gomes	Universidade Nova de Lisboa, Portugal
Hani Hagras	University of Essex, UK
Takahiro Hara	Osaka University, Japan
Houcine Hassan	Polytechnical University of Valencia, Spain
Naohiro Hayashibara	Tokyo Denki University, Japan
Pin-Han Ho	University of Waterloo, Canada
Pao-Ann Hsiung	National Chung Cheng University, Taiwan

Program/Technical Committee (continued)

Chung-hsing Hsu	Los Alamos National Laboratory, USA
Yueh-Min Huang	National Cheng Kung University, Taiwan
Chung-Ming Huang	National Cheng Kung University, Taiwan
Jason C. Hung	Kung Wu Institute of Technology, Taiwan
Hoh Peter In	Korea University, Korea
Pedro Isaias	Portuguese Open University, Portugal
Kenichi Ito	Siebold University of Nagasaki, Japan
Rong-Hong Jan	National Chiao Tung University, Taiwan
Qun Jin	Waseda University, Japan
Mahmut Kandemir	Pennsylvania State University, USA
Jien Kato	Nagoya University, Japan
Daeyoung Kim	Information and Communications University, Korea
Akio Koyama	Yamagata University, Japan
Christian Landrault	LIRMM, France
Trong-Yen Lee	National Taipei University of Technology, Taiwan
Yannick Le Moullec	Aalborg University, Denmark
Regis Leveugle	INPG/CSI, France
Xiaoming Li	Peking University, China
Yiming Li	National Chiao Tung University, Taiwan
Zhiyuan Li	Purdue University, USA
Minglu Li	Shanghai Jiaotong University, China
Wen-Hwa Liao	Tatung University, Taiwan
Shih-wei Liao	INTEL, USA
Man Lin	St. Francis Xavier University, Canada
Youn-Long Lin	National Tsing Hua University, Taiwan
Alex Zhaoyu Liu	University of North Carolina at Charlotte, USA
Lucia Lo Bello	University of Catania, Itlay
Renato Lo Cigno	University of Trento, Italy
Jianhua Ma	Hosei University, Japan
Petri Mahonen	Aachen University, Germany
Juan-Miguel Martinez	Polytechnical University of Valencia, Spain
Pedro M. Ruiz Martinez	University of Murcia, Spain
Geyong Min	University of Bradford, UK
Marius Minea	Politehnica University of Timisoara, Romania
Daniel Mosse	University of Pittsburgh, USA
Takuo Nakashima	Kyushu Tokai University, Japan
Amiya Nayak	University of Ottawa, Canada
Joseph Ng	Hong Kong Baptist University, Hong Kong
Sala Nicoletta	University of Italian Switzerland, Switzerland
Gianluca Palermo	Politecnico di Milano, Italy

Program/Technical Committee (continued)

Vassilis Paliouras	University of Patras, Greece
Raju Pandey	University of California at Davis, USA
Preeti Panta	Indian Institute of Technology, India
Massimo Poncino	Politecnico di Torino, Italy
Isabelle Puaut	University of Rennes, France
Sanjay Rajopadhye	Colorado State University, USA
Maurizio Rebaudengo	Politecnico di Torino, Italy
Xiangshi Ren	Kochi University of Technology, Japan
Achim Rettberg	University of Paderborn, Germany
Bikash Sabata	IET Inc., USA
Takamichi Saito	Meiji University, Japan
Biplab K. Sarker	University of New Brunswick, Canada
Fumiaki Sato	Shizuoka University, Japan
Klaus Schneider	University of Kaiserslautern, Germany
Berhard Scholz	University of Sydney, Australia
Win-Bin See	Aerospace Industrial Development, Taiwan
Jaume Segura	University of Illes Balears, Spain
Weisong Shi	Wayne State University, USA
Timothy K. Shih	Tamkang University, Taiwan
Kuei-Ping Shih	Tamkang University, Taiwan
Dimitrios Soudris	Democritus University of Thrace, Greece
Robert Steele	University of Technology Sydney, Australia
Takuo Suganuma	Tohoku University, Japan
Kaoru Sugita	Fukuoka Institute of Technology, Japan
Walid Taha	Rice University, USA
David Taniar	Monash University, Australia
Tsutomu Terada	Osaka University, Japan
Eduardo Tovar	Instituto Politecnico do Porto, Portugal
Yu-Chee Tseng	National Chiao Tung University, Taiwan
Hung-ying Tyan	National Sun Yat-sen University, Taiwan
Tom Vander Aa	IMEC, Belgium
Luminita Vasiu	University of Westminster, UK
Diego Vazquez	Centro Nacional de Microelectronica, Spain
Jari Veijalainen	University of Jyvaskyla, Finland
Salvatore Venticinque	Second University of Naples, Italy
Arnaud Virazel	LIRMM, France
Salvatore Vitabile	University of Palermo, Italy
Natalija Vlajic	York University, Canada
Guojun Wang	Central South University, China
Cho-Li Wang	The University of Hong Kong, China
Frank Zhigang Wang	Cranfield University, UK
Hengshan Wang	University of Shanghai for Science and Technology, China

Program/Technical Committee (continued)

Xingwei Wang	Northeastern University, China
Allan Wong	Hong Kong Polytechnic University, China
Jie Wu	Florida Atlantic University, USA
Shih-Lin Wu	Chang Gung University, Taiwan
Chenyong Wu	Chinese Academy of Sciences, China
Zhaohui Wu	Zhejiang University, China
Hans-Joachim Wunderlich	University of Stuttgart, Germany
Bin Xiao	Hong Kong Polytechnic University, China
Chengzhong Xu	Wayne State University, USA
Chu-Sing Yang	National Sun Yat-sen University, Taiwan
Jianhua Yang	Dalian University of Technology, China
Hongji Yang	De Montfort University, UK
Tomokazu Yoneda	Nara Institute of Science and Technology, Japan
Muhammad Younas	Oxford Brookes University, UK
Sergio Yovine	IMAG, France
Gwo-Jong Yu	Aletheia University, Taiwan
Qing-An Zeng	University of Cincinnati, USA
Hongbin Zha	Peking University, China
Chaohai Zhang	Kumamoto University, Japan
Jingyuan Zhang	University of Alabama, USA
Shengbing Zhang	Northwestern Polytechnical University, China
Yi Zhang	University of Electronic Science and Technology of China, China
Yongbing Zhang	University of Tsukaba, Japan
Youtao Zhang	University of Texas at Dallas, USA
Weiming Zheng	Tsinghua University, China
Aoying Zhou	Fudan University, China
Chunguang Zhou	Jilin University, China
Xiaobo Zhou	University of Colorado at Colorado Springs, USA
Dakai Zhu	University of Texas at San Antonio, USA
Hao Zhu	Florida International University, USA

Additional Reviewers

Gian-Franco Dalla Betta	Antoine Gallais	Danilo Severina
Damiano Carra	Mark Halpern	Wei Wang
Valentina Casola	Mauro Iacono	
Oliver Diessel	Stefano Marrone	

Table of Contents

Keynote

Nanotechnology in the Service of Embedded and Ubiquitous Computing
Niraj K. Jha 1

Parallel Embedded Systems: Optimizations and Challenges
Edwin H.-M. Sha 2

Progress of Ubiquitous Information Services and Keeping Their Security by Biometrics Authentication
Kazuo Asakawa 3

Embedded Hardware

Implementing and Evaluating Color-Aware Instruction Set for Low-Memory, Embedded Video Processing in Data Parallel Architectures
Jongmyon Kim, D. Scott Wills, Linda M. Wills 4

A DSP-Enhanced 32-Bit Embedded Microprocessor
Hyun-Gyu Kim, Hyeong-Cheol Oh 17

An Intelligent Sensor for Fingerprint Recognition
Salvatore Vitabile, Vincenzo Conti, Giuseppe Lentini, Filippo Sorbello 27

Exploiting Register-Usage for Saving Register-File Energy in Embedded Processors
Wann-Yun Shieh, Chien-Chen Chen 37

Hardware Concurrent Garbage Collection for Short-Lived Objects in Mobile Java Devices
Chi Hang Yau, Yi Yu Tan, Anthony S. Fong, Wing Shing Yu 47

An Effective Instruction Cache Prefetch Policy by Exploiting Cache History Information
Soong Hyun Shin, Cheol Hong Kim, Chu Shik Jhon 57

Efficient Switches for Network-on-Chip Based Embedded Systems
Hsin-Chou Chi, Chia-Ming Wu 67

An Efficient Dynamic Switching Mechanism (DSM) for Hybrid Processor Architecture
Akanda Md. Musfiquzzaman, Ben A. Abderazek, Sotaro Kawata, Masahiro Sowa 77

Design of Face Recognition Door Manager System Based on DSP
Dongbing Pu, Changrui Du, Zhezhou Yu, Chunguang Zhou 87

Embedded Software

AlchemistJ: A Framework for Self-adaptive Software
Dongsun Kim, Sooyong Park 98

Design and Implementation of Accounting System for Information Appliances
Midori Sugaya, Shuichi Oikawa, Tatsuo Nakajima 110

Loop Distribution and Fusion with Timing and Code Size Optimization for Embedded DSPs
Meilin Liu, Qingfeng Zhuge, Zili Shao, Chun Xue, Meikang Qiu, Edwin H.-M. Sha 121

Ensuring Real-Time Performance Guarantees in Dynamically Reconfigurable Embedded Systems
Aleksandra Tešanović, Mehdi Amirijoo, Daniel Nilsson, Henrik Norin, Jörgen Hansson 131

ANTS: An Evolvable Network of Tiny Sensors
Daeyoung Kim, Tomás Sánchez López, Seongeun Yoo, Jongwoo Sung, Jaeeon Kim, Youngsoo Kim, Yoonmee Doh 142

Design Models for Reusable and Reconfigurable State Machines
Christo Angelov, Krzysztof Sierszecki, Nicolae Marian 152

Optimizing Nested Loops with Iterational and Instructional Retiming
Chun Xue, Zili Shao, Meilin Liu, Meikang Qiu, Edwin H.-M. Sha 164

Real-Time Systems

Realtime H.264 Encoding System Using Fast Motion Estimation and Mode Decision
Byeong-Doo Choi, Min-Cheol Hwang, Jun-Ki Cho, Jin-Sam Kim, Jin-Hyung Kim, Sung-Jea Ko 174

Polyhedra-Based Approach for Incremental Validation of Real-Time Systems
David Doose, Zoubir Mammeri 184

Checkpointing for the Reliability of Real-Time Systems with On-Line Fault Detection
Sang-Moon Ryu, Dong-Jo Park 194

Parallelizing Serializable Transactions Within Distributed Real-Time Database Systems
Subhash Bhalla, Masaki Hasegawa 203

Timing Analysis of Distributed End-to-End Task Graphs with Model-Checking
Zonghua Gu 214

Power-Aware Computing

Voronoi-Based Improved Algorithm for Connected Coverage Problem in Wireless Sensor Networks
Jie Jiang, Zhen Song, Heying Zhang, Wenhua Dou 224

Near Optimal and Energy-Efficient Scheduling for Hard Real-Time Embedded Systems
Amjad Mohsen, Richard Hofmann 234

Performance Evaluation of Power-Aware Communication Network Devices
Hiroyuki Okamura, Tadashi Dohi 245

An Energy Aware, Cluster-Based Routing Algorithm for Wireless Sensor Networks
Jyh-Huei Chang, Rong-Hong Jan 255

Energy-Constrained Prefetching Optimization in Embedded Applications
Juan Chen, Yong Dong, Hui-zhan Yi, Xue-jun Yang 267

An Energy Reduction Scheduling Mechanism for a High-Performance SoC Architecture
Slo-Li Chu 281

H/S Co-design and Systems-on-Chip

A New Buffer Planning Algorithm Based on Room Resizing
Hongjie Bai, Sheqin Dong, Xianlong Hong, Song Chen 291

Analyzing the Performance of Mesh and Fat-Tree Topologies for Network on Chip Design
Vu-Duc Ngo, Huy-Nam Nguyen, Hae-Wook Choi 300

Hierarchical Graph: A New Cost Effective Architecture for Network on Chip
Alireza Vahdatpour, Ahmadreza Tavakoli, Mohammad Hossein Falaki 311

RISC/DSP Dual Core Wireless SoC Processor Focused on Multimedia Applications
Hyo-Joong Suh, Jeongmin Kim 321

An Accurate Architectural Simulator for ARM1136
Hyo-Joong Suh, Sung Woo Chung 331

Modular Design Structure and High-Level Prototyping for Novel Embedded Processor Core
Ben A. Abderazek, Sotaro Kawata, Tsutomu Yoshinaga, Masahiro Sowa 340

Pipelined Bidirectional Bus Architecture for Embedded Multimedia SoCs
Gang-Hoon Seo, Won-Yong Jung, Seongsoo Lee, Jae-Kyung Wee 350

On Tools for Modeling High-Performance Embedded Systems
Anilkumar Nambiar, Vipin Chaudhary 360

A Hardware/Software Co-design and Co-verification on a Novel Embedded Object-Oriented Processor
Chi Hang Yau, Yi Yu Tan, Pak Lun Mok, Wing Shing Yu, Anthony S. Fong 371

Testing and Verification

Timed Weak Simulation Verification and Its Application to Stepwise Refinement of Real-Time Software
Satoshi Yamane 381

Checking Component-Based Embedded Software Designs for Scenario-Based Timing Specifications
Jun Hu, Xiaofeng Yu, Yan Zhang, Tian Zhang, Xuandong Li, Guoliang Zheng 395

Dependable Polygon-Processing Algorithms for Safety-Critical Embedded Systems
Jens Brandt, Klaus Schneider 405

Reconfigurable Computing

New Area Management Method Based on "Pressure" for Plastic Cell Architecture
Taichi Nagamoto, Satoshi Yano, Mitsuru Uchida, Yuichiro Shibata, Kiyoshi Oguri 418

Evaluation of Space Allocation Circuits
Shinya Kyusaka, Hayato Higuchi, Taichi Nagamoto, Yuichiro Shibata, Kiyoshi Oguri 428

Automatic Configuration with *Conflets*
Justinian Oprescu, Franck Rousseau, Andrzej Duda 438

Path Concepts for a Reconfigurable Bit-Serial Synchronous Architecture
Florian Dittmann, Achim Rettberg, Raphael Weber 448

An FPGA-Based Parallel Accelerator for Matrix Multiplications in the Newton-Raphson Method
Xizhen Xu, Sotirios G. Ziavras, Tae-Gyu Chang 458

A Run-Time Partitioning Algorithm for RTOS on Reconfigurable Hardware
Marcelo Götz, Achim Rettberg, Carlos Eduardo Pereira 469

UML-Based Design Flow and Partitioning Methodology for Dynamically Reconfigurable Computing Systems
Chih-Hao Tseng, Pao-Ann Hsiung 479

Hardware Task Scheduling and Placement in Operating Systems for Dynamically Reconfigurable SoC
Yuan-Hsiu Chen, Pao-Ann Hsiung 489

Agent and Distributed Computing

An Intelligent Agent for RFID-Based Home Network System
Woojin Lee, Juil Kim, Kiwon Chong 499

An Intelligent Adaptation System Based on a Self-growing Engine
Jehwan Oh, Seunghwa Lee, Eunseok Lee 509

Dynamically Selecting Distribution Strategies for Web Documents According to Access Pattern
Wenyu Qu, Di Wu, Keqiu Li, Hong Shen 518

Web-Based Authoring Tool for e-Salesman System
Magdalene P. Ting, Jerry Gao 528

Agent-Community-Based P2P Semantic Web Information Retrieval System Architecture
Haibo Yu, Tsunenori Mine, Makoto Amamiya 538

A Scalable and Reliable Multiple Home Regions Based Location Service in Mobile Ad Hoc Networks
Guojun Wang, Yingjun Lin, Minyi Guo 550

Global State Detection Based on Peer-to-Peer Interactions
Punit Chandra, Ajay D. Kshemkalyani 560

Nonintrusive Snapshots Using Thin Slices
Ajay D. Kshemkalyani, Bin Wu 572

Load Balanced Allocation of Multiple Tasks in a Distributed Computing System
Biplab Kumer Sarker, Anil Kumar Tripathi, Deo Prakash Vidyarthi, Laurence Tianruo Yang, Kuniaki Uehara 584

Wireless Communications

ETRI-QM: Reward Oriented Query Model for Wireless Sensor Networks
Jie Yang, Lei Shu, Xiaoling Wu, Jinsung Cho, Sungyoung Lee, Sangman Han 597

Performance of Signal Loss Maps for Wireless Ad Hoc Networks
Henry Larkin, Zheng da Wu, Warren Toomey 609

Performance Analysis of Adaptive Mobility Management in Wireless Networks
Myung-Kyu Yi 619

A Novel Tag Identification Algorithm for RFID System Using UHF
Ho-Seung Choi, Jae-Hyun Kim 629

Coverage-Aware Sensor Engagement in Dense Sensor Networks
Jun Lu, Lichun Bao, Tatsuya Suda 639

A Cross-Layer Approach to Heterogeneous Interoperability in Wireless Mesh Networks
Shih-Hao Shen, Jen-Wen Ding, Yueh-Min Huang 651

Reliable Time Synchronization Protocol for Wireless Sensor Networks
Soyoung Hwang, Yunju Baek 663

HMNR Scheme Based Dynamic Route Optimization to Support Network Mobility of Mobile Network
Moon-Sang Jeong, Jong-Tae Park, Yeong-Hun Cho 673

QoS Routing with Link Stability in Mobile Ad Hoc Networks
Jui-Ming Chen, Shih-Pang Ho, Yen-Cheng Lin, Li-Der Chou 683

Mobile Computing

Efficient Cooperative Caching Schemes for Data Access in Mobile Ad Hoc Networks
Cheng-Ru Young, Ge-Ming Chiu, Fu-Lan Wu 693

Supporting SIP Personal Mobility for VoIP Services
Tsan-Pin Wang, KauLin Chiu 703

Scalable Spatial Query Processing for Location-Aware Mobile Services
KwangJin Park, MoonBae Song, Ki-Sik Kong, Chong-Sun Hwang, Kwang-Sik Chung, SoonYoung Jung 715

Exploiting Mobility as Context for Energy-Efficient Location-Aware Computing
MoonBae Song, KwangJin Park, Ki-Sik Kong 725

Mobile User Data Mining: Mining Relationship Patterns
John Goh, David Taniar 735

Asymmetry-Aware Link Quality Services in Wireless Sensor Networks
Junzhao Du, Weisong Shi, Kewei Sha 745

Incorporating Global Index with Data Placement Scheme for Multi Channels Mobile Broadcast Environment
Agustinus Borgy Waluyo, Bala Srinivasan, David Taniar, Wenny Rahayu 755

An Adaptive Mobile Application Development Framework
Ming-Chun Cheng, Shyan-Ming Yuan 765

Multimedia, HCI and Pervasive Computing

Context-Aware Emergency Remedy System Based on Pervasive Computing
Hsu-Yang Kung, Mei-Hsien Lin, Chi-Yu Hsu, Chia-Ni Liu 775

Design and Implementation of Interactive Contents Authoring Tool for MPEG-4
Hsu-Yang Kung, Che-I Wu, Jiun-Ju Wei 785

A Programmable Context Interface to Build a Context Infrastructure for Worldwide Smart Applications
Kyung-Lang Park, Chang-Soon Kim, Chang-Duk Kang, Shin-Dug Kim 795

Adaptive Voice Smoothing with Optimal Playback Delay Based on the ITU-T E-Model
Shyh-Fang Huang, Eric Hsiao-Kuang Wu, Pao-Chi Chang 805

The Wearable Computer as a Personal Station
Jin Ho Yoo, Sang Ho Lee 816

Perception of Wearable Computers for Everyday Life by the General Public: Impact of Culture and Gender on Technology
Sébastien Duval, Hiromichi Hashizume 826

Videoconference System by Using Dynamic Adaptive Architecture for Self-adaptation
Chulho Jung, Sanghee Lee, Eunseok Lee 836

Contextual Interfacing: A Sensor and Actuator Framework
Kasper Hallenborg 846

MDR-Based Framework for Sharing Metadata in Ubiquitous Computing Environment
O-Hoon Choi, Jung-Eun Lim, Doo-Kwon Baik 858

Design of System for Multimedia Streaming Service
Chang-Soo Kim, Hag-Young Kim, Myung-Joon Kim, Jae-Soo Yoo . . . 867

A Workflow Language Based on Structural Context Model for Ubiquitous Computing
Joohyun Han, Yongyun Cho, Jaeyoung Choi 879

Ubiquitous Content Formulations for Real-Time Information Communications
K.L. Eddie Law, Sunny So 890

A Semantic Web-Based Infrastructure Supporting Context-Aware Applications
Renato F. Bulcão-Neto, Cesar A.C. Teixeira, Maria da Graça C. Pimentel 900

A Universal PCA for Image Compression
Chuanfeng Lv, Qiangfu Zhao 910

An Enhanced Ontology Based Context Model and Fusion Mechanism
Yingyi Bu, Jun Li, Shaxun Chen, Xianping Tao, Jian Lv 920

A Framework for Video Streaming to Resource-Constrained Terminals
Dmitri Jarnikov, Johan Lukkien, Peter van der Stok 930

Fragile Watermarking Scheme for Accepting Image Compression
Mi-Ae Kim, Kil-Sang Yoo, Won-Hyung Lee 940

PUML and PGML: Device-Independent UI and Logic Markup Languages on Small and Mobile Appliances
Tzu-Han Kao, Yung-Yu Chen, Tsung-Han Tsai, Hung-Jen Chou, Wei-Hsuan Lin, Shyan-Ming Yuan 947

Distributed Contextual Information Storage Using Content-Centric Hash Tables
Ignacio Nieto, Juan A. Botía, Pedro M. Ruiz, Antonio F. Gómez-Skarmeta 957

An Integrated Scheme for Address Assignment and Service Location in Pervasive Environments
Mijeom Kim, Mohan Kumar, Behrooz Shirazi 967

Modeling User Intention in Pervasive Service Environments
Pascal Bihler, Lionel Brunie, Vasile-Marian Scuturici 977

The Performance Estimation of the Situation Awareness RFID System from Ubiquitous Environment Scenario
Dongwon Jeong, Heeseo Chae, Hoh Peter In 987

The Content Analyzer Supporting Interoperability of MPEG-4 Content in Heterogeneous Players
Hyunju Lee, Sangwook Kim 996

Adaptive Voice Smoother with Optimal Playback Delay for New Generation VoIP Services
Shyh-Fang Huang, Eric Hsiao-Kuang Wu, Pao-Chi Chang 1006

Designing a Context-Aware System to Detect Dangerous Situations in School Routes for Kids Outdoor Safety Care
Katsuhiro Takata, Yusuke Shina, Hiraku Komuro, Masataka Tanaka, Masanobu Ide, Jianhua Ma 1016

An Advanced Mental State Transition Network and Psychological Experiments
Peilin Jiang, Hua Xiang, Fuji Ren, Shingo Kuroiwa 1026

Development of a Microdisplay Based on the Field Emission Display Technology
Takahiro Fusayasu, Yoshito Tanaka, Kazuhiko Kasano, Hisashi Fukuda, Peisong Song, Bongi Kim 1036

Network Protocol, Security and Fault-Tolerance

Information Flow Security for Interactive Systems
Ying Jin, Lei Liu, Xiao-juan Zheng 1045

A Microeconomics-Based Fuzzy QoS Unicast Routing Scheme in NGI
Xingwei Wang, Meijia Hou, Junwei Wang, Min Huang 1055

Considerations of Point-to-Multipoint QoS Based Route Optimization Using PCEMP
Dipnarayan Guha, Seng Kyoun Jo, Doan Huy Cuong, Jun Kyun Choi 1065

Lightweight Real-Time Network Communication Protocol for Commodity Cluster Systems
Hai Jin, Minghu Zhang, Pengliu Tan, Hanhua Chen, Li Xu 1075

Towards a Secure and Reliable System
Michele Portolan, Régis Leveugle 1085

Optimal Multicast Loop Algorithm for Multimedia Traffic Distribution
Yong-Jin Lee, M. Atiquzzaman 1099

An Effective Method of Fingerprint Classification Combined with AFIS
Ching-Tang Hsieh, Shys-Rong Shyu, Chia-Shing Hu 1107

A Hierarchical Anonymous Communication Protocol for Sensor Networks
Arjan Durresi, Vamsi Paruchuri, Mimoza Durresi, Leonard Barolli 1123

A Network Evaluation for LAN, MAN and WAN Grid Environments
Evgueni Dodonov, Rodrigo Fernandes de Mello, Laurence Tianruo Yang 1133

SVM Classifier Incorporating Feature Selection Using GA for Spam Detection
Huai-bin Wang, Ying Yu, Zhen Liu 1147

Middleware and P2P Computing

Adaptive Component Allocation in ScudWare Middleware for Ubiquitous Computing
Qing Wu, Zhaohui Wu 1155

Prottoy: A Middleware for Sentient Environment
Fahim Kawsar, Kaori Fujinami, Tatsuo Nakajima 1165

Middleware Architecture for Context Knowledge Discovery in Ubiquitous Computing
Kim Anh Ngoc Pham, Young Koo Lee, Sung Young Lee 1177

Ubiquitous Computing: Challenges in Flexible Data Aggregation
Eiko Yoneki, Jean Bacon 1189

Author Index 1201

Nanotechnology in the Service of Embedded and Ubiquitous Computing

Niraj K. Jha

Department of Electrical Engineering,
Princeton University, Princeton,
NJ 08544, USA
jha@ee.princeton.edu

Embedded systems are now ubiquitous and ubiquitous computing is now getting embedded in our day-to-day lives. Such systems are cost and power sensitive. Various nanotechnologies will provide an excellent vehicle to reduce cost and power consumption, while still meeting performance constraints. Nanoscale device technologies, such as carbon nanotube transistors, nanowires, resonant-tunneling devices, quantum cellular automata, single electron transistors, tunneling phase logic, and a host of others, have made significant advances in the last few years. However, circuit and system design methodologies for these technologies are still in their infancy. Industrial roadmaps project that these emergent technologies will make inroads in the commercial market within a decade. Therefore, such design methodologies are necessary for precise design and fabrication of nanocircuits and nanoarchitectures.

In this talk, we will try to bring together the three exciting disciplines of embedded systems, ubiquitous computing and nanotechnology. We will show how various nanotechnologies may be of service to embedded and ubiquitous computing.

In many nanotechnologies, the basic logic primitive is a threshold gate or a majority/minority gate. Using traditional logic design methods for such technologies is inadequate. Testing and defect tolerance techniques for such technologies will also merit special consideration. To meet the challenges of low-cost/low-power computing in the coming decade, we will need analysis and synthesis tools at all levels of the system design hierarchy. We will discuss initial efforts in this area and speculate on how merging of nanotechnology with embedded/ubiquitous computing can be brought about.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, p. 1, 2005.
© IFIP International Federation for Information Processing 2005

Parallel Embedded Systems: Optimizations and Challenges*

Edwin H.-M. Sha

University of Texas at Dallas,
Richardson, Texas 75083, USA
edsha@utdallas.edu

With the advance of system level integration and system-on-chip, the high-tech industry is now moving toward multiple-core parallel embedded systems using hardware/software co-design approach. To design and optimize an embedded system and its software is technically hard because of the strict requirements of an embedded system in timing, code size, memory, low power, security, etc. while optimizing a parallel embedded system makes research even more challenging. The research in embedded systems needs integrated efforts in many areas such as algorithms, computer architectures, compilers, parallel/distributed processing, real-time systems, etc.

This talk will first use an example to illustrate how to find the best parallel algorithm and architecture for this example application, and the technical challenges on design of parallel embedded systems. Because loops are usually the most critical parts to be optimized in DSP or any computation-intensive applications, the talk will then present our results in various types of optimizations for loops in timing, code-size, memory, power consumption, heterogeneous systems, etc. Many of our techniques give the best known results available in literatures. This talk will show that using our multi-dimensional retiming technique, any uniform nested loops can be transformed such that all the computations in the new loop body can be executed simultaneously. This is the best possible result and can be applied to many applications executed on VLIW or other types of parallel systems.

* This work is partially supported by TI University Program, NSF EIA-0103709, Texas ARP 009741-0028-2001, NSF CCF-0309461, NSF IIS-0513669, Microsoft, USA.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, p. 2, 2005.
© IFIP International Federation for Information Processing 2005

Progress of Ubiquitous Information Services and Keeping Their Security by Biometrics Authentication

Kazuo Asakawa

Fujitsu Laboratories, Ltd.,
4-1-1 Kamikodanaka, Nakahara-ku,
Kawasaki 211-8588, Japan
asakawa@labs.fujitsu.com

With progress of both fixed and mobile networks, various ubiquitous information services have been brought up to apply in actual business scene. In fact, it becomes possible to receive and send information in various access points not only in offices but also in cars through sophisticated ubiquitous terminals such as PDA, cellular phone, mobile PC and so on. Moreover, the spread of RFID-tag application to household commodities will accelerate the popularization of ubiquitous services. Every ubiquitous terminal will have RFID reader/writer on board in near future. By using the ubiquitous information service environment, variety of services such as e-government, e-bank, e-commerce, Intelligent Transport Systems service that usually require customized services for each customer has already been provided. However, high-tech crimes such as skimming and phishing are increasing day by day inversely increment of its convenience. Such high-tech crimes were happened in worldwide actually. It is impossible to keep security by 4 or more digits PIN code any longer. Biometrics authentication is promising to protect against high-tech crimes in the present situation. I will give an overview of current ubiquitous information services, focusing on some of actual services in keeping security by biometrics authentication.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, p. 3, 2005.
© IFIP International Federation for Information Processing 2005

Implementing and Evaluating Color-Aware Instruction Set for Low-Memory, Embedded Video Processing in Data Parallel Architectures*

Jongmyon Kim[1], D. Scott Wills[2], and Linda M. Wills[2]

[1] Chip Solution Center, Samsung Advanced Institute of Technology,
San 14-1, Nongseo-ri, Kiheung-eup, Kyungki-do, 449-712, South Korea
jongmyon.kim@samsung.com
[2] School of Electrical and Computer Engineering,
Georgia Institute of Technology, Atlanta, Georgia 30332-0250
{scott.wills, linda.wills}@ece.gatech.edu

Abstract. Future embedded imaging applications will be more demanding processing performance while requiring the same low cost and low energy consumption. This paper presents and evaluates a color-aware instruction set extension (CAX) for single instruction, multiple data (SIMD) processor arrays to meet the computational requirements and cost goals. CAX supports parallel operations on two-packed 16-bit (6:5:5) YCbCr data in a 32-bit datapath processor, providing greater concurrency and efficiency for color image and video processing. Unlike typical multimedia extensions (e.g., MMX, VIS, and MDMX), CAX harnesses parallelism within the human perceptual color space rather than depending solely on generic subword parallelism. Moreover, the ability to reduce data format size reduces system cost. The reduction in data bandwidth also simplifies system design. Experimental results on a representative SIMD array architecture show that CAX achieves a speedup ranging from 5.2× to 8.8× (an average of 6.3×) over the baseline SIMD array performance. This is in contrast to MDMX (a representative MIPS multimedia extension), which achieves a speedup ranging from 3× to 5× (an average of 3.7×) over the same baseline SIMD array. CAX also outperforms MDMX in both area efficiency (a 52% increase versus a 13% increase) and energy efficiency (a 50% increase versus an 11% increase), resulting in better component utilization and sustainable battery life. Furthermore, CAX improves the performance and efficiency with a mere 3% increase in the system area and a 5% increase in the system power, while MDMX requires a 14% increase in the system area and a 16% increase in the system power. These results demonstrate that CAX is a suitable candidate for application-specific embedded multimedia systems.

1 Introduction

As multimedia revolutionizes our society, its applications are becoming some of the most dominant computing workloads. Color image and video processing in particular

* This work was performed by authors at the Georgia Institute of Technology (Atlanta, GA).

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 4 – 16, 2005.
© IFIP International Federation for Information Processing 2005

has garnered considerable interest over the past few years since color features are valuable in sensing the environment, recognizing objects, and conveying crucial information [10]. These applications, however, demand tremendous computational and I/O throughput. Moreover, increasing user demand for multimedia-over-wireless capabilities on embedded systems places additional constraints on power, size, and weight.

Single instruction, multiple data (SIMD) architectures have demonstrated the potential to meet the computational requirements and cost goals by employing thousands of inexpensive processing elements (PEs) and distributing and co-locating PEs with the data I/O to minimize storage and data communication requirements. The SIMD Pixel (SIMPil) processor [2, 4], for example, is a low memory, monolithically integrated SIMD architecture that efficiently exploits massive data parallelism inherent in imaging applications. It reduces data movement through a processing-in-place technique in which image data are directly transported into the PEs and stored there. Two-dimensional SIMD arrays, including SIMPil, are well suited for many imaging tasks that require processing of pixel data with respect to either nearest-neighbor or other 2-D patterns exhibiting locality or regularity. However, they are less amenable to vector (multichannel) processing in which each pixel computation is performed simultaneously on 3-D YCbCr (luminance-chrominance) channels [1], which are widely used in the image and video processing community. More specifically, since the 3-D vector computation is performed within innermost loops, its performance does not scale with larger PE arrays.

This paper presents a color-aware instruction set extension (CAX) for such SIMD arrays as a solution to this performance limitation by supporting two-packed 16-bit (6:5:5) YCbCr data in a 32-bit register, while processing these color data in parallel. (CAX was introduced previously for superscalar processors [7], but this paper is investigating its use in SIMD image processing architectures.) The YCbCr space allows coding schemes that exploit the properties of human vision by truncating some

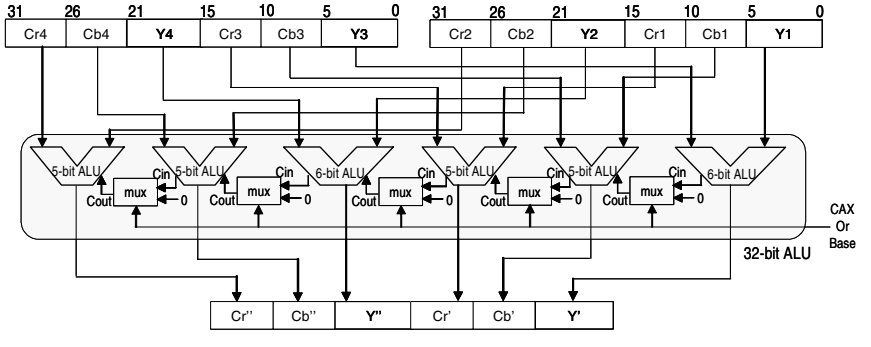


Fig. 1. An example of a partitioned ALU functional unit that exploits color subword parallelism

of the less important data in every color pixel and allocating fewer bits to the high-frequency chrominance components that are perceptually less significant. Thus, it provides satisfactory image quality in a compact 16-bit color representation that consists of a six-bit luminance (Y) and two five-bit chrominance (Cr and Cb)

components [6]. In addition, CAX offers greater concurrency with minimal hardware modification. Fig. 1 shows an example of how a 32-bit ALU functional unit can be used to perform either a 32-bit baseline ALU or two 6:5:5-bit ALUs. The 32-bit ALU is divided into two six-bit ALUs and four five-bit ALUs. When the output carry (Cout) is blocked (i.e., Cin = 0), the six smaller ALUs can be performed in parallel.

This paper evaluates CAX in comparison to a representative multimedia extension, MIPS MDMX [11], in a specified SIMD array architecture. MDMX was chosen as a basis of comparison because it provides an effective way of dealing with reduction operations, using a wide packed accumulator that successively accumulates the results produced by operations on multimedia vector registers. Other multimedia extensions (e.g., Intel MMX [9] and Sun VIS [14]) provide more limited support of vector processing in a 32-bit datapath processor without accumulators. To handle vector processing on a 64-bit or 128-bit datapath, they require frequent packing/unpacking of operand data, deteriorating their performance.

This evaluation shows that CAX outperforms MDMX in performance and efficiency metrics on the same SIMD array because CAX benefits from greater concurrency and reduced pixel word storage which can consume a large percentage of silicon area. In particular, the key findings are the following. MDMX achieves a speedup ranging from 3× to 5× (an average of 3.7×) over the baseline performance. However, MDMX requires a 14% increase in the system area and a 16% increase in the system power. As a result, MDMX improves energy efficiency from only 2% to 24% and area efficiency from 6% to 22% over the baseline. On the other hand, CAX achieves a speedup ranging from 5.2× to 8.8× (an average of 6.3×) over the baseline performance because of greater subword parallelism. Moreover, the higher performance is achieved with a mere 3% increase in the system area and a 5% increase in the system power. As a result, CAX improves area efficiency from 36% to 68% and energy efficiency from 35% to 77% over the baseline. These results demonstrate that CAX provides an efficient mechanism for embedded imaging systems.

The rest of the paper is organized as follows. Section 2 presents a summary of the CAX instruction set. Section 3 describes the modeled architectures and a methodology infrastructure for the evaluation of CAX. Section 4 evaluates the system area and power of our modeled architectures, and Section 5 analyzes execution performance and efficiency for each case. Section 6 concludes this paper.

2 Color-Aware Instruction Set for Color Imaging Applications

The color-aware instruction set (CAX) efficiently eliminates the computational burden of vector processing by supporting parallel operations on two-packed 16-bit (6:5:5) YCbCr data in a 32-bit datapath processor. In addition, CAX employs a 128-bit color-packed accumulator that provides a solution to overflow and other issues caused by packing data as tightly as possible by implicit width promotion and adequate space. Fig. 2 illustrates three types of operations: (1) a baseline 32-bit operation, (2) a 4 × 8-bit SIMD operation used in many general-purpose processors, and (3) a 2 × 16-bit CAX operation employing heterogeneous (non-uniform) subword parallelism.

For color images, the band data may be interleaved (e.g., the red, green, and blue data of each pixel are adjacent in memory) or separated (e.g., the red data for adjacent pixels are adjacent in memory). Although the band separated format is the most convenient for SIMD processing, a significant amount of overhead for data alignment is expected prior to SIMD processing. Moreover, traditional SIMD data communication operations have trouble with the band data that are not aligned on boundaries that are powers of two (e.g., adjacent pixels from each band are visually spaced three bytes apart) [12]. Even if the SIMD multimedia extensions store the pixel information in the band-interleaved format (i.e., |Unused|R|G|B| in a 32-bit register), subword parallelism cannot be exploited on the operand of the unused field. Furthermore, since the RGB space does not model the perceptual attributes of human vision well, the RGB to YCbCr conversion is required prior to color image processing. Although the SIMD multimedia extensions can handle the color conversion process in software, the hardware approach would be more efficient.

CAX solves problems inherent to packed RGB extensions by properly aligning two-packed 16-bit data on 32-bit boundaries and by directly supporting YCbCr data processing, providing greater concurrency and efficiency for processing color image sequences. The CAX instructions are classified into four different groups: (1) parallel arithmetic and logical instructions, (2) parallel compare instructions, (3) permute instructions, and (4) special-purpose instructions.

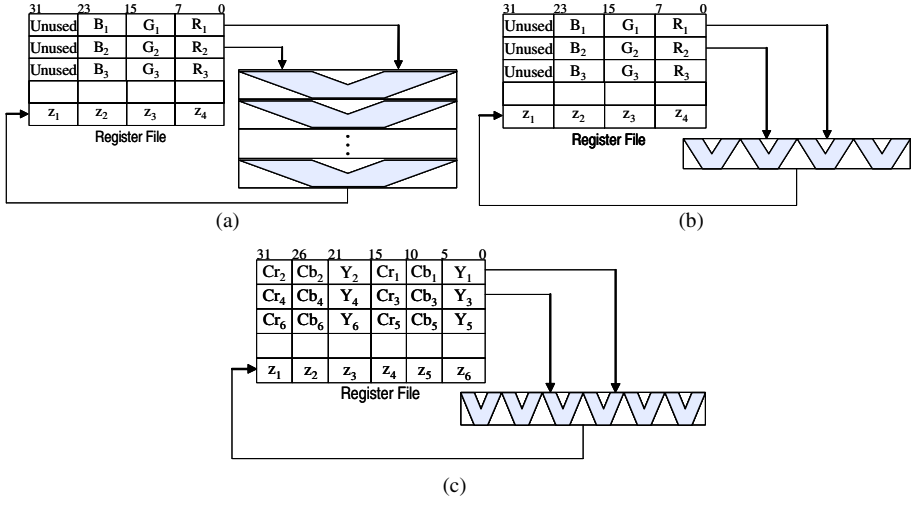


Fig. 2. Types of operations: (a) a baseline 32-bit operation, (b) a 32-bit SIMD operation, and (c) a 32-bit CAX operation

2.1 Parallel Arithmetic and Logical Instructions

Parallel arithmetic and logical instructions include packed versions of addition (`ADD_CRCBY`), subtraction (`SUBTRACT_CRCBY`), and average (`AVERAGE_CRCBY`). The addition and subtraction instructions include a saturation operation that clamps

the output result to the largest or smallest value for the given data type when an overflow occurs. Saturating arithmetic is particularly useful in pixel-related operations, for example, to prevent a black pixel from becoming white if an overflow occurs. The parallel average instruction, which is useful for blending algorithms, takes two packed data types as input, adds corresponding data quantities, and divides each result by two while placing the result in the corresponding data location. The rounding is performed to ensure precision over repeated average instructions.

2.2 Parallel Compare Instructions

Parallel compare instructions include `CMPEQ_CRCBY`, `CMPNE_CRCBY`, `CMPGE_CRCBY`, `CMPGT_CRCBY`, `CMPLE_CRCBY`, `CMPLT_CRCBY`, `CMOV_CRCBY` (conditional move), `MIN_CRCBY`, and `MAX_CRCBY`. These instructions compare pairs of sub-elements (e.g., Y, Cb, and Cr) in the two source registers. Depending on the instructions, the results are varied for each sub-element comparison. The first seven instructions are useful for a condition query performed on the incoming data such as chroma-keying [9]. The last two instructions, `MIN_CRCBY` and `MAX_CRCBY`, are especially useful for median filtering, which compare pairs of sub-elements in the two source registers while outputting the minimum and maximum values to the target register.

2.3 Parallel Permute Instructions

Permute instructions include `MIX_CRCBY` and `ROTATE_CRCBY`. These instructions are used to rearrange the order of quantities in the packed data type. The mix instruction mixes the sub-elements of the two source registers into the operands of the target register, and the rotate instruction rotates the sub-elements to the right by an immediate value. These instructions are useful for performing a vector pixel transposition or a matrix transposition [13].

2.4 Special-Purpose Instructions

Special-purpose CAX instructions include `ADACC_CRCBY` (absolute-differences-accumulate), `MACC_CRCBY` (multiply-accumulate), `RAC` (read accumulator), and `ZACC` (zero accumulator), which provide the most computational benefits of all the CAX instructions. The `ADACC_CRCBY` instruction, for example, is frequently used in a number of algorithms for motion estimation. The `MACC_CRCBY` instruction is useful in DSP algorithms that involve computing a vector dot-product, such as digital filters and convolutions. The last two instructions `RAC` and `ZACC` are related to the managing of the CAX accumulator.

3 Methodology

This section describes modeled architectures and a methodology infrastructure for the evaluation of the CAX instruction set.

3.1 Modeled Architectures

The SIMD Pixel (SIMPil) processor is used as the baseline SIMD image processing architecture for this study. Fig. 3 shows the microarchitecture of the SIMD array, along with its interconnection network. When data are distributed, the processing elements (PEs) execute a set of instructions in a lockstep fashion. With 4×4 pixel sensor sub-arrays, each PE is associated with a specific portion (4×4 pixels or 16 pixel-per-processing-element) of an image frame, allowing streaming pixel data to be retrieved and processed locally. Each PE has a reduced instruction set computer (RISC) datapath with the following minimum characteristics:

- Small amount of local storage (128 32-bit words),
- Three-ported general-purpose registers (16 32-bit words),
- ALU – computes basic arithmetic and logic operations,
- Barrel shifter – performs multi-bit logic/arithmetic shift operations,
- MACC – multiplies 32-bit values and accumulates into a 64-bit accumulator,
- Sleep – activates or deactivates a PE based on local information,
- Pixel unit – samples pixel data from the local image sensor array,
- ADC unit – converts light intensities into digital values,
- RGB2YCC and YCC2RGB unit– converts RGB to/from YCbCr, and
- Nearest neighbor communications through a NEWS (north-east-west-south) network and serial I/O unit.

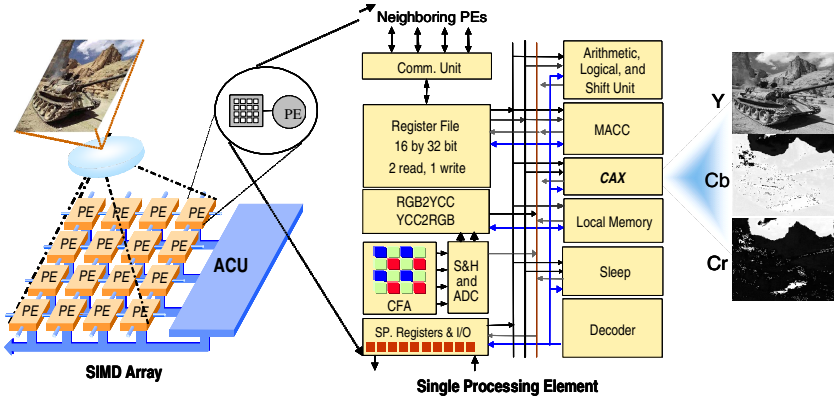


Fig. 3. Block diagram of a SIMD array and a processing element

To improve the performance of vector processing of color image sequences, CAX instructions are included in the instruction set architecture (ISA) of the SIMPil array. For a performance comparison, MDMX-type instructions are also included in the SIMPil ISA. Table 1 summarizes the parameters of the modeled architectures. An overall simulation infrastructure is presented next.

Table 1. Modeled architecture parameters

Parameter	Value
System Size	44×38 (1,584 PEs)
Image Sensor per PE (vertor pixel per PE ratio)	4×4 (16 VPPE)
VLSI Technology	100 nm
Clock Frequency	80 MHz
Interconnection Network	Mesh
intALU/intMUL/Barrel Shifter/intMACC/Comm	1 / 1 / 1 / 1 / 1
MDMX/CAX: intALU/intMACC	1 / 1
Local Memory Size (baseline/MDMX/CAX)	128 32-bit/ 128 32-bit/ 64 32-bit word

3.2 Methodology Infrastructure

Fig. 4 shows a methodology infrastructure that is divided into three levels: application, architecture, and technology.

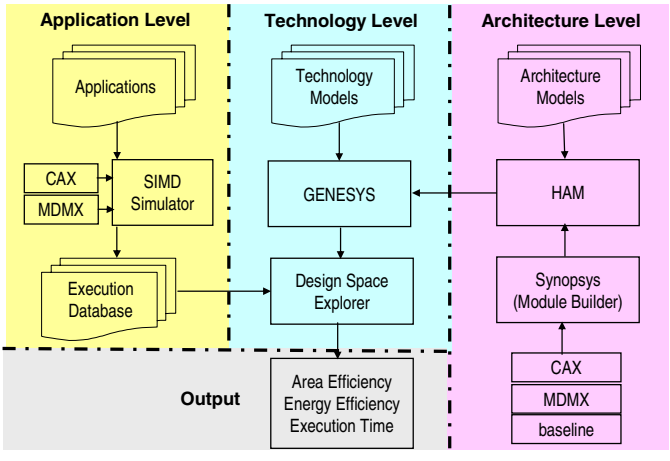


Fig. 4. A methodology infrastructure for exploring the design space of three modeled architectures: baseline SIMPil, MDMX-SIMPil, and CAX-SIMPil

At the application level, an instruction-level SIMD simulator, called SIMPilSim, has been used to profile execution statistics, such as cycle count, dynamic instruction frequency, and PE utilization, for the three different versions of the programs: (1) baseline ISA without subword parallelism (SIMPil), (2) baseline plus MDMX ISA (MDMX-SIMPil), and (3) baseline plus CAX ISA (CAX-SIMPil). The benchmark suite includes five imaging applications (see more details at [5]): a chroma-keying program (CHROMA), color edge detection using a vector Sobel operator (VSobel), the vector median filter (VMF), vector quantization (VQ), and the full-search vector block-matching algorithm of motion estimation (FSVBMA) within the MPEG standard.

At the architecture level, the heterogeneous architectural modeling (HAM) of functional units for SIMD arrays, proposed by Chai *et al.* [3], has been used to calculate the design parameters of modeled architectures. For the design parameters of the MDMX and CAX functional units (FUs), Verilog models for the baseline, MDMX, and CAX FUs were implemented and synthesized with the Synopsys design compiler (DC) using a 0.18-micron standard cell library. The reported area specifications of the MDMX and CAX FUs were then normalized to the baseline FU, and the normalized numbers were applied to the HAM tool for determining the design parameters of MDMX- and CAX-SIMPil. The design parameters are then passed to the technology level.

At the technology level, the Generic System Simulator (GENESYS) developed at Georgia Tech [8] has been used to calculate technology parameters (e.g., latency, area, power, and clock frequency) for each configuration. Finally, the databases (e.g., cycle times, instruction latencies, instruction counts, area, and power of the functional units) obtained from the application, architecture, and technology levels are combined to determine execution times, area efficiency, and energy efficiency for each case. The next section presents the system area and power of the modeled architectures.

4 System Area and Power Evaluation Using Technology Modeling

Fig. 5 shows the system area and power of MDMX-SIMPil and CAX-SIMPil, normalized to the baseline SIMPil. Experimental results indicate that MDMX requires a 14% increase in the entire system area and a 16% increase in the peak system power. However, CAX only requires a 3% increase in the system area and a 5% increase in the system area and power because of the reduced pixel word storage (local memory). These system area and power results are combined with application simulations for determining processing performance, area efficiency, and energy efficiency for each case, which is presented next.

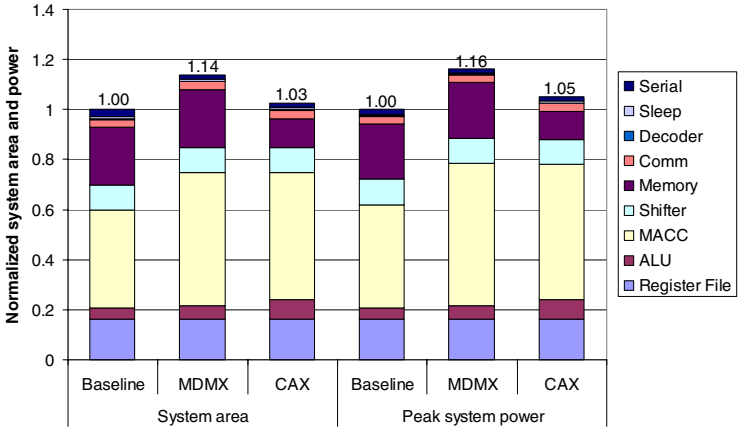


Fig. 5. System area and power of CAX-SIMPil and MDMX-SIMPil, normalized to the baseline SIMPil

5 Experimental Results

Cycle accurate simulation and technology modeling have been used to determine the performance and efficiency characteristics of modeled architectures for each application task. Each application was developed in its respective assembly languages for the SIMPil system, in which all three versions for each program have the same parameters, data sets, and calling sequences. In the experiment, the overhead of the color conversion was not included in the performance evaluation for all the versions. In other words, this study assumes that the baseline, MDMX, and CAX versions directly support YCbCr data in the band-interleaved format (e.g., |Unused|Cr|Cb|Y| for baseline and MDMX and |Cr|Cb|Y|Cr|Cb|Y| for CAX). The metrics of the execution cycle count, corresponding sustained throughput, energy efficiency, and area efficiency of each case form the basis of the study comparison, defined in Table 2.

Table 2. Summary of evaluation metrics

execution time	sustained throughput	area efficiency	energy efficiency
$t_{exec} = \frac{C}{f_{ck}}$	$Th_{sust} = \frac{O_{exec} \cdot U \cdot N_{PE}}{t_{exec}}$	$\eta_A = \frac{Th_{sust}}{Area}[\frac{\text{Gops}}{\text{s} \cdot \text{mm}^2}]$	$\eta_E = \frac{O_{exec} \cdot U \cdot N_{PE}}{Energy}[\frac{\text{Gops}}{\text{Joule}}]$

C is the cycle count, f_{ck} is the clock frequency, O_{exec} is the number of executed operations, U is the system utilization, and N_{PE} is the number of processing elements. Note that since each CAX and MDMX instruction executes more operations (typically six and three times, respectively) than a baseline instruction, we assume that each CAX, MDMX, and baseline instruction executes six, three, and one operation, respectively, for the sustained throughput calculation.

5.1 Performance Evaluation Results

This section evaluates the impact of CAX on processing performance for the selected color imaging applications on the SIMPil system.

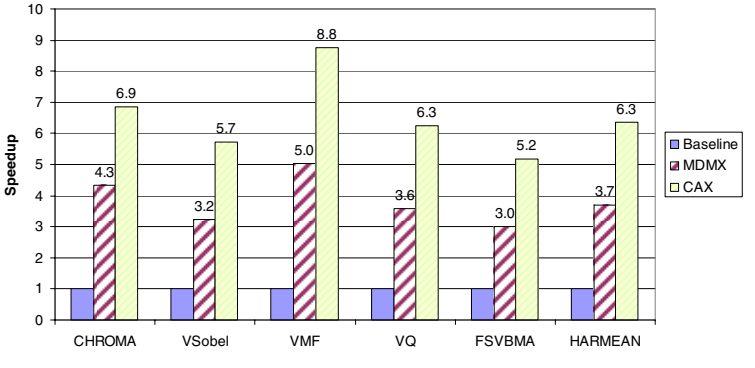


Fig. 6. Speedups for the SIMPil system with CAX and MDMX, normalized to the baseline performance. Note that HARMEAN is the harmonic mean.

Overall Results. Fig. 6 illustrates execution performance (speedups in executed cycles) attained by CAX and MDMX when compared with the baseline performance without subword parallelism. The results indicate that CAX outperforms MDMX for all the programs in terms of speedup, indicating a speedup ranging from 5.2× to 8.8× (an average of 6.3×) with CAX, but only 3× to 5× (an average of 3.7×) with MDMX over the baseline. The next section discusses the sources for the reductions in the issued instructions are discussed next.

Benefits of CAX for Color Imaging Applications. Fig. 7 shows the distribution of issued vector instructions for the SIMPil system with CAX and MDMX, normalized to the baseline version. Each bar divides the instructions into the arithmetic-logic-unit (ALU), memory (MEM), communication (COMM), PE activity control unit (MASK), image pixel loading (PIXEL), MDMX, and CAX. The use of CAX reduces a significant number of the instruction counts for all of the programs, ranging from 88.6% (VMF) to 80.7% (FSVBMA) over the baseline. In particular, CAX reduces a significant number of ALU and memory instruction counts due to its instruction definition. An interesting observation is that the FSVBMA program has the smallest reduction in the instruction count with CAX. This is because it involves high inter-PE communication operations that are not affected by CAX. For example, each PE cannot directly support a macroblock size of 16×16 pixels because 4×4 pixels are mapped to each PE. As a result, the 4×4 distortions are computed in each PE separately. Each result is then combined through NEWS communication instructions for the final distortion between the 16×16 input and reference blocks.

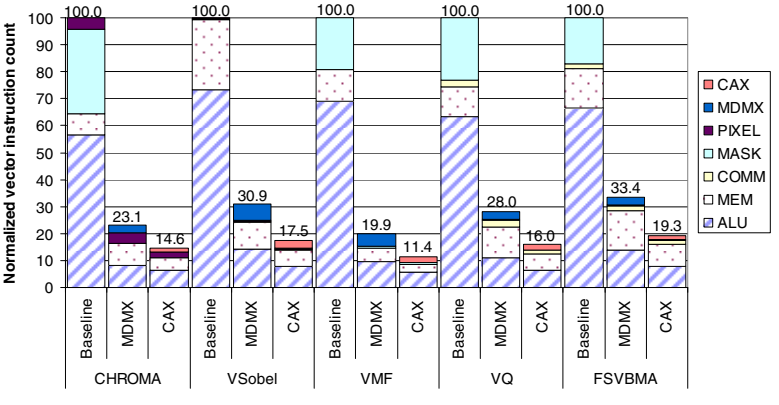


Fig. 7. The distribution of issued vector instructions for the SIMPil system with CAX and MDMX, normalized to the baseline version

5.2 Energy Evaluation Results

Fig. 8 shows energy efficiency, the task throughput achieved per unit of Joule, for the SIMPil system with MDMX and CAX, normalized to the baseline version. CAX outperforms MDMX across all the programs in the energy efficiency, indicating a

50% increase with CAX, but only an 11% increase with MDMX. This is because CAX achieves higher sustained throughputs with a smaller increase in the system power. Increasing energy efficiency improves sustainable battery life for given system capabilities.

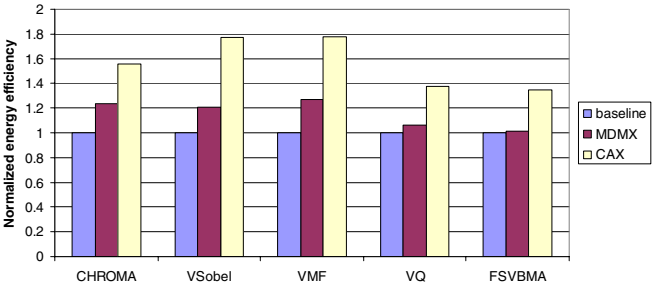


Fig. 8. Energy efficiency for the SIMPil system with CAX and MDMX, normalized to the baseline version

5.3 Area Evaluation Results

Fig. 9 shows area efficiency, the task throughput achieved per unit of area, for the SIMPil system with MDMX and CAX, normalized to the baseline version. As with energy efficiency, CAX outperforms MDMX for all the programs in the area efficiency, indicating a 52% increase with CAX, but only a 13% increase with MDMX. This is because CAX achieves higher sustained throughput with smaller area overhead. Increasing area efficiency improves component utilization for given system capabilities.

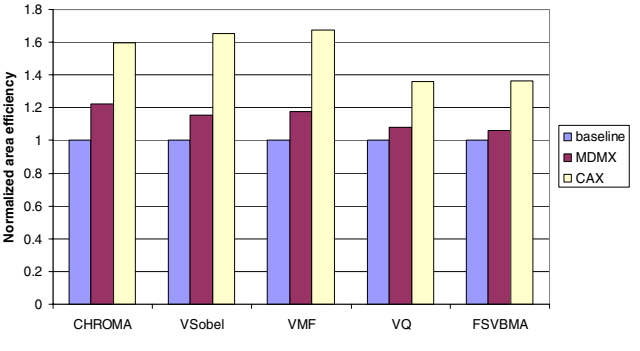


Fig. 9. Area efficiency for the SIMPil system with CAX and MDMX, normalized to the baseline version

6 Conclusions

As emerging portable multimedia applications demand more and more tremendous computational throughput with limited area and power, the need for high efficiency, high throughput embedded processing is becoming an important challenge in computer architecture. In this regard, this paper has addressed application-, architecture-, and technology-level issues in an existing processing system to efficiently support vector processing of color image sequences. In particular, this paper has focused on the color-aware instruction set (CAX) for memory- and performance-hungry embedded applications in a representative SIMD image processing architecture. Unlike typical multimedia extensions, CAX harnesses parallelism within the human perceptual color space (e.g., YCbCr). Rather than depending solely on generic subword parallelism, CAX supports parallel operations on two-packed 16-bit YCbCr data in a 32-bit datapath processor, providing greater concurrency and efficiency for color image and video processing. The key findings are as follows:

- CAX achieves a speedup ranging from 5.2× to 8.8× (an average of 6.3×) over the baseline SIMD array performance without subword parallelism. This is in contrast to MDMX, which achieves a speedup ranging from only 3× to 5× (an average of 3.7×) over the same baseline SIMD array.
- CAX reduces energy consumption from 80% to 89%, but MDMX reduces energy consumption from only 60% to 79% over the baseline version.
- Moreover, CAX benefits from reduced pixel word storage in addition to greater concurrency. As a result, CAX outperforms MDMX for all the programs in area efficiency and energy efficiency. The area efficiency increases from 36% to 68% (an average of 52%) with CAX, but only 6% to 22% (an average of 13%) with MDMX. The energy efficiency increases from 35% to 77% (an average of 50%) with CAX, but only 2% to 24% (an average of 11%) with MDMX. Increasing area and energy efficiencies yield greater component utilization and sustainable battery life, respectively, for given system capabilities.
- Furthermore, CAX improves the performance and efficiency with a mere 3% increase in the silicon area and a 5% increase in the system power, while MDMX requires a 14% increase in the silicon area and a 16% increase in the system power.

In the future, a heuristic compiler support will be explored that extracts both data-level parallelism and color subword parallelism from high level language programs to overcome tedious hand optimization and/or special programming libraries.

References

1. V. Bhaskaran and K. Konstantinides, Image and Video Compression Standards: Algorithms and Architectures, Kluwer Academic Publishers (1997)
2. H. H. Cat, A. Gentile, J. C. Eble, M. Lee, O. Verdier, Y. J. Joo, D. S. Wills, M. Brooke, N. M. Jokerst, A. S. Brown, and R. Leavitt, SIMPil: An OE integrated SIMD architecture for focal plane processing applications, in Proc. Massively Parallel Processing Using Optical Interconnection (MPPOI-96), pp. 44-52 (1996)

3. S. M. Chai, T. M. Taha, D. S. Wills, and J. D. Meindl, Heterogeneous architecture models for interconnect-motivated system design, IEEE Trans. VLSI Systems, special issue on system level interconnect prediction, vol. 8, no. 6, pp. 660-670 (2000)
4. A. Gentile and D. S. Wills, Portable Video Supercomputing, IEEE Trans. on Computers, vol. 53, no. 8, pp. 960-973 (2004)
5. J. Kim, Architectural enhancements for color image and video processing on embedded systems. PhD dissertation, Georgia Inst. of Technology (2005)
6. J. Kim and D. S. Wills, Evaluating a 16-bit YCbCr (6:5:5) color representation for low memory, embedded video processing, in Proc. of the IEEE Intl. Conf. on Consumer Electronics, pp. 181-182 (2005)
7. J. Kim and D. S. Wills, Efficient processing of color image sequences using a color-aware instruction set on mobile systems, in Proc. of the IEEE Intl. Conf. on Application-Specific Systems, Architectures, and Processors, pp. 137-149 (2004)
8. S. Nugent, D. S. Wills, and J. D. Meindl, A hierarchical block-based modeling methodology for SoC in GENESYS, in Proc. of the 15th Ann. IEEE Intl. ASIC/SOC Conf., pp. 239-243 (2002)
9. A. Peleg and U. Weiser, MMX technology extension to the Intel architecture, IEEE Micro, vol.16, no. 4, pp. 42-50 (1996).
10. K. N. Plataniotis and A. N. Venetsanopoulos, Color Image Processing and Applications (2000)
11. MIPS extension for digital media with 3D. Technical Report http://www.mips.com, MIPS technologies, Inc. (1997)
12. N. Slingerland and A. J. Smith, Measuring the performance of multimedia instruction sets, IEEE Trans. on Computers, vol. 51, no. 11, pp. 1317-1332 (2002)
13. J. Suh and V. K. Prasanna, An efficient algorithm for out-of-core matrix transposition, IEEE Trans. on Computers, vol. 51, no. 4, pp. 420-438 (2002)
14. M. Tremblay, J. M. O'Connor, V. Narayanan, and L. He, VIS speeds new media processing, IEEE Micro, vol. 16, no. 4, pp. 10-20 (1996)

A DSP-Enhanced 32-Bit Embedded Microprocessor

Hyun-Gyu Kim[1,2] and Hyeong-Cheol Oh[3]

[1] Dept. of Elec. and Info. Eng., Graduate School, Korea Univ., Seoul 136-701, Korea
[2] R&D center, Advanced Digital Chips, Seoul 135-508, Korea
babyworm@gmail.com
[3] Dept. of Info. Eng., Korea Univ. at Seo-Chang. Chung-Nam 339-700, Korea
ohyeong@korea.ac.kr

Abstract. EISC (Extendable Instruction Set Computer) is a compressed code architecture developed for embedded applications. In this paper, we propose a DSP-enhanced embedded microprocessor based on the 32-bit EISC architecture. We present how we could exploit the special features, and how we could overcome the deficits, of the EISC architecture to accelerate DSP applications with a relatively low hardware overhead. Our simulations and experiments show that the proposed DSP-enhanced processor reduces the average execution time of the DSP kernels considered in this work by 47.8% and the DSP applications by 29.3%. The proposed DSP enhancements cost about 10300 gates and do not increase the clock frequency. The proposed DSP-enhanced processor has been embedded in an SoC for video processing and proven in silicon.

Keywords: DSP-enhanced microprocessor, SIMD, hardware address generator, register extension, embedded microprocessor.

1 Introduction

As more and more DSP (digital signal processing) applications run on embedded systems in recent years, it has become one of the most important tasks for embedded microprocessors to accelerate DSP applications. This trend is being reflected on the most successful embedded processors in the market: ARM cores added saturation arithmetic in ARMv5TE and SIMD (Single Instruction Multiple Data) instruction set in ARMv6 [1]; and MIPS cores adopted an application-specific-extension instruction set for DSP and 3D applications [2]. The capabilities of accelerating DSP accelerations should be implemented with as little hardware overhead as possible, since most embedded microprocessors target on the cost and power sensitive markets.

In this paper, we introduce a DSP-enhanced embedded microprocessor based on the EISC (Extendable Instruction Set Computer) architecture [3, 4, 5]. EISC is a compressed code architecture developed for embedded applications. While achieving high code density and a low memory access rate, the EISC architecture uses a novel and terse scheme to resolve the problem of insufficient immediate

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 17–26, 2005.
© IFIP International Federation for Information Processing 2005

operand fields of the compressed code RISC machines. We present how we could exploit the special features of the EISC architecture to accelerate DSP applications with a relatively low hardware overhead. Since EISC also has deficits in processing DSP applications like any other compressed code architectures, we propose various schemes to overcome these deficits of the EISC.

In order to seek proper enhancements, we analyze the workload of benchmark programs from Mediabench [6] on the 32-bit EISC processor, called *base* below. The processor *base* is a 5-stage pipelined implementation of the non-DSP 32-bit instruction set of EISC [5]. Based on the profiling results, we modify the *base* by carefully choosing and adding DSP supporting instructions, such as SIMD MAC and saturation arithmetic. We also adopt and add various schemes for signal processing, such as a support for correcting radix point during the fixed-point multiplications, a hardware address generator for supporting memory accesses efficiently, and a scheme for increasing the effective number of general-purpose registers (GPRs).

Our simulations and experiments show that the proposed DSP-enhanced processor reduces the average execution time of the DSP kernels considered in this work by 47.8% and the DSP applications by 29.3%. The proposed DSP enhancements cost approximately 10300 gates only and do not increase the clock frequency.

This paper is organized as follows: In Section 2, an overview of the EISC architecture is presented. Section 3 describes the schemes we propose for enhancing DSP capabilities. Section 4 presents our evaluation results and an SoC that has been developed based on the proposed processor. Section 5 concludes the paper.

2 The EISC Architecture

In an embedded microprocessor system, code density and chip area are two major design issues, since these costs are more important in this area. However, many 32-bit embedded microprocessors suffer from poor code density. In order to address this problem, some RISC-based 32-bit microprocessors adopt 16-bit compressed instruction set architectures, such as ARM THUMB [7] and MIPS16 [8]. This approach provides better code density but needs some mechanisms to extend the insufficient immediate field and to provide backward compatibility with their previous architectures, which can result in extra hardware overhead. Moreover, these architectures have difficulty in utilizing their registers efficiently in the compressed mode [7, 8].

The EISC architecture takes a different approach for achieving high code density [3, 4]. EISC uses efficient fixed length 16-bit instruction set for 32-bit data processing. To resolve the problem of insufficient immediate operand fields in a terse way, EISC uses an independent instruction called `leri`, which consists of a 2-bit opcode and a 14-bit immediate value. The `leri` instruction loads immediate value to the special register called ER (extension register), and the

value in the ER is used for extending the immediate field of a related instruction to make a long immediate operand.

By using the `leri` instruction, the EISC architecture can make the program code more compact than the competing architectures, ARM-THUMB and MIPS16, since the frequency of the `leri` instruction is much less than 20% in most programs. In [3], the code density of the EISC architecture was evaluated to be 6.5% higher than that of ARM THUMB and 11.5% higher than that of MIPS16 for various programs considered in [3]. In our experiments, the static code sizes of *base* are 18.9% smaller than those of ARM9-TDMI for the programs in Mediabench. As the `leri` instruction is just used to extend the immediate field of a related instruction, it can be a performance burden for the EISC processor. To overcome this deficit, EISC uses *leri instruction folding method* explained in [10].

As EISC uses fixed length 16-bit instruction set, the instruction decoder for the EISC architecture is much simpler than that of the CISC(Complex Instruction Set Computer) architecture. In addition, the EISC does not suffer from the overhead for switching its processor mode, which is often needed to handle long immediate values or complicated instructions in the compressed code RISC architectures. Moreover, the EISC architecture reduces its data memory access rate by fully utilizing its 16 registers while the competing architectures can access limited number of registers in the compressed mode. In [4], the data memory access rate of the EISC is 35.1% less than that of the ARM THUMB and 37.6% less than that of MIPS16 for the programs considered in [4]. Thus, the EISC can reduce both instruction references and data references. Reducing the memory accesses would bring reduction in power consumption related to the memory accesses and also lessen the performance degradation caused by the speed gap between the processor and the memory.

3 DSP Enhancements for the EISC Processor

In this paper, we propose several DSP enhancements for developing a DSP-enhanced EISC processor with as little extra hardware cost as possible. We develop instructions for enhancing DSP performances: instructions for SIMD operations with the capability of saturation arithmetic; and instructions for generating addresses and packing, loading, and storing media data. We try to minimize the overheads for feeding data into the SIMD unit [11]. We develop the enhancements so that they can be realized within the limited code space, since the processor uses 16-bit instructions (even though it has a 32-bit datapath.)

3.1 DSP Instruction Set

Since the data in the multimedia applications are frequently smaller than a word, we can boost up the performance of the processor *base* by adopting a SIMD architecture. When the SIMD operations are performed, however, some expensive packing operations are also performed for arranging the data to the

SIMD operation unit [11]. In order to reduce the number of packing operations, the processor should support various types of packed data. However, since the code space allowed for the DSP-enhancements is limited as mentioned above, we focus on accelerating the MAC operations which are apparently the most popular operations in the DSP applications. We implement two types of MAC operations shown in Fig. 1: the parallel SIMD type and the sum-of-products type. The parallel SIMD type of MAC operation, shown in Fig. 1(a), is much more efficient than any other types of operations for processing the arrays of structured data such as RGBA formatted pixel data. On the other hand, the sum-of-products type of MAC operation, shown in Fig. 1(b), is efficient for processing general data arrays.

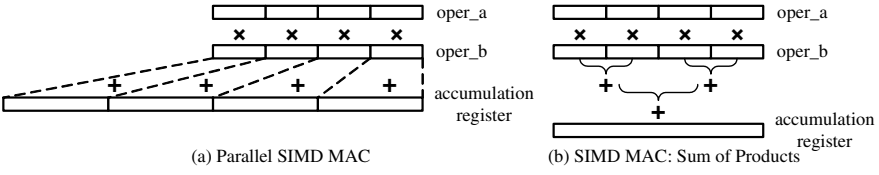


Fig. 1. Two types of the SIMD MAC operations supported in the proposed DSP-enhanced processor

We also adopt the instructions for the saturation arithmetic which is often used in DSP applications. Unlike the wrap-around arithmetic, the saturation arithmetic uses the maximum or the minimum value when it detects an overflow or an underflow.

In the signal processing, the fixed-point arithmetic is commonly used because it is much cheaper and faster than the floating-point arithmetic. During the multiplication of two 32-bit fixed-point numbers, the results are stored in the 64-bit multiplication result register (MR). Since the position of radix point in a fixed-point number can be changed during multiplications, we need to select a part of the 64-bit multiplication result to make a 32-bit fixed-point number. For that purpose, *base* would require a sequence of five instructions looking like the one in the first box shown below. We propose to use an instruction with the mnemonic `mrs` (*multiplication result selection*), as shown below.

```
mfmh    %Rx           # move MR(H) to GPR(Rx)
mfml    %Ry           # move MR(L) to GPR(Ry)
asl     DP, %Rx       # DP-bit shift left Rx
lsr     (32-DP), %Ry  # (32-DP)-bit shift right Ry
or      %Rx, %Ry      # Ry = Rx | Ry
```

↓

```
mrs     DP, %Ry
```

3.2 Accelerating Address Generation

DSP applications usually perform a set of computations repeatedly for streaming data that have almost uniform data patterns. In this section, we introduce a loop-efficient hardware address generator that can accelerate the memory-addressing processes. We intend to use this address generator for accelerating DSP applications, including the memory-copy operations, with low cost.

The proposed address generator has a capability of handling the auto-increment addressing mode which is usually used for various applications including the memory-copy operations. As DSP operations use a loop for computing processes to handle streaming data, the proposed address generation unit is equipped with a comparator to detect the end-offset for loop. The address generator is also intended for supporting other special memory-addressing modes, such as the wrap-around incremental addressing mode for a virtual circular buffer and the bit-reversal addressing mode for transform kernels. We only support the post-modifying scheme because it is more popularly used [12].

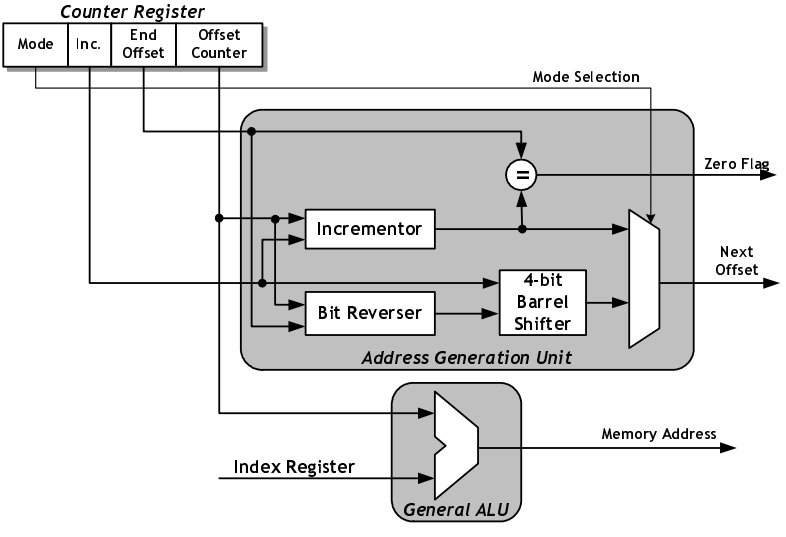


Fig. 2. Proposed address generation unit

Since the hardware complexity is important, the proposed address generator only produces the small sized offset instead of the entire addresses. Fig. 2 shows the block diagram of the proposed address generator and the counter register. The control register is used to control the address generator and holds a control word regarding generation pattern, offset counter, end offset for looping, and incremental offset. Since we use a post-modifying scheme, the ALU uses the previously generated offset counter to make a memory address, while the incrementor uses it to generate the next offset. The generated offset is written back to the offset counter field in the counter register.

3.3 Register Extension

Almost DSP routines use many coefficients and temporary values simultaneously. However, most embedded microprocessors do not have enough registers to hold those values. A general approach for resolving this problem of shortage of registers is to spill the contents of registers into the memory. However, media data often take the form of data streams, in which case they are temporarily stored in the memory, continuously read into the processor, and used only a few times. Therefore, media applications running on an embedded processor often suffer from significant performance losses due to the frequent register spilling processes.

We propose a register extension scheme that increases the effective number of registers by adopting the idea of shadow register to hold temporary values in the registers. The idea of shadow register has been used in various aspects, such as the context switching latency [13]. We find that this idea is also very useful for embedded processors which have limited code spaces, such as the EISC processors, since it is hard for them to allocate additional bits for register indexing [9].

For the register extension scheme, we divide the register file into two parts: one part consists of a set of smaller register files, *register pages*, and another part consists of eight registers to be shared among register pages. Each register page consists of eight registers. We use three bits in the status register to select active register page. Thus we can use up to seventy two registers, eight shared registers and eight register pages with eight registers, in the proposed processor when whole register pages are implemented. The use of register pages decreases memory traffic due to memory spilling.

4 Evaluations

The designed DSP-enhanced microprocessor has been modeled in Verilog HDL as a synthesizable RTL model. We synthesize the model using the Samsung STD130 0.18μm CMOS standard cell library [14] and the Synopsys DesignCompiler. The critical path delay is estimated by using the Synopsys PrimeTime under the worst-case operating condition (1.65V supply voltage and 125 °C junction temperature.) The results are summarized in Table 1. As shown in Table 1, the critical path delay is almost the same even through some units are added to accelerate DSP applications. The enhancements we add cost about 10270 equivalent gates, most of which are used for the shadow register file and the SIMD MAC unit. The others cost less than 1000 gates per each unit.

In order to evaluate the performance of the proposed architecture, we use DSP kernels used commonly in many DSP applications: IMDCT (inverse modified discrete cosine transform), FFT (fast Fourier transform), DCT (discrete cosine transform), and FIR (finite impulse response) filter. The IMDCT routine is used for reducing aliases in high fidelity audio coding applications including MP3, and AAC. The DCT is commonly used in the image compression. The FFT and FIR

Table 1. Cost and power performance of the enhancements

	Area [equi. gates]	Critial Path Delay [ns]
Base processor	56852	6.25
Proposed processor	67122	6.19

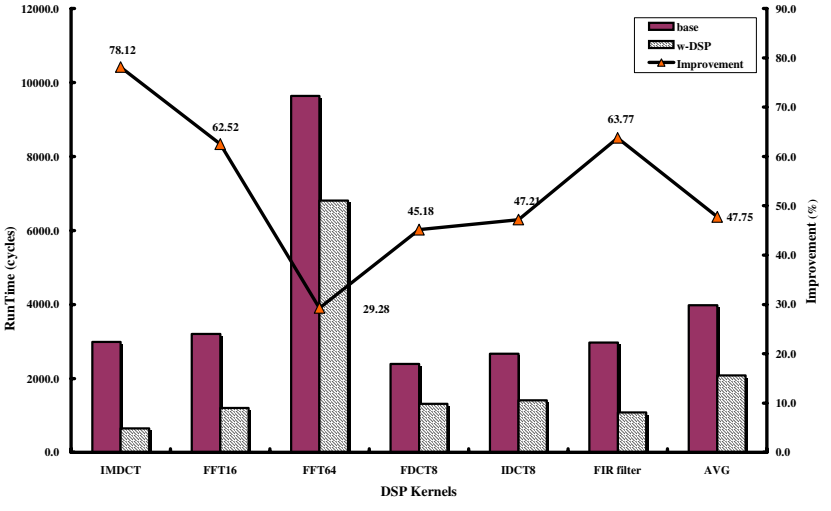


Fig. 3. Performance gain for popular DSP kernels

filter are the most common DSP functions in many DSP applications. We use the verilog-HDL model with the perfect memory model during the performance evaluation.

The results of the experiments are summarized in Fig. 3. The proposed DSP architecture reduces the average execution time of the DSP kernels considered by 47.8%. For IMDCT, the use of the `mac` and `mrs` results in the reductions of the computation time. Furthermore, the use of the register extension scheme reduces the number of memory accesses by 9%. For FFT kernels, the performance improvement is mainly due to the parallel SIMD MAC operations. However, we observe a limited performance enhancement for the 64-point FFT, since it takes too much time to process the data packing and the memory accesses for spilling temporary values. For DCT and FIR filter, the instruction for sum of products is used efficiently and results in the performance gain. Moreover, the register extension scheme reduces memory operations in the FIR filter application. In our experiments, the coefficients for an 8-TAP FIR filter are loaded just once during the processing.

We also experiment with real DSP applications such as MP3 decoding program which uses MAD (MPEG audio decoding) library, ADPCM (Adaptive

Differential Pulse Code Modulation) encoding program, and JPEG image decoding program. While selecting the functions to be optimized, we analyze the DSP applications to identify functions which take a large portion of the run time. In the experiments, a perfect (zero-wait) external memory is also assumed. The input data that we use and the selected functions are summarized in Table 2.

Table 2. Input data and selected kernels for the considered DSP applications

DSP application	Input data	Selected kernel(s)
MP3 decoding	8.124-second stereo MP3 sample, 44.1KHz sampling rate, 192bps bit rate	*imdct_l* *subband synthesis*
JPEG decoding	64 × 64 × 24b jpeg file	*Huffman decode*
ADPCM encoding	3.204-second mono PCM sample, 8KHz sampling rate, 16-bit little endian	*ADPCM encode*

We measure the clock counts. The results are summarized in Table 3. For MP3 decoding application, optimized *imdct_l* and *subband synthesis* kernels reduce the run time by 30.8%. It means that the proposed DSP-enhanced processor is able to decode a high-fidelity MP3 audio at 31.1MHz, while the processor *base* has to be clocked at 45MHz to perform the same job. While optimizing the JPEG decoder, `cntl` instructions and SIMD saturate add instructions are used to optimize *Huffman decode* and the functions related on the variable length coding. In the case of the ADCPM encoding application, we reduce the execution time by 36.9% using the address generator and saturate add instructions. As a result, the proposed processor reduces the average execution time of the DSP applications considered by 29.3%.

Table 3. Performance gain for DSP applications

DSP applications	*Base processor*	*Proposed processor*	Improvement
MP3 decoding	365,827,110	252,923,593	30.9%
JPEG decoding	4,507,509	3,597,074	20.2%
ADPCM encoding	1,914,368	1,208,188	36.9%

The proposed DSP-enhanced processor has been embedded in an SoC for video processing [15] and proven in silicon. Fig. 4 shows the layout of the SoC. The SoC is equipped with a 2D graphics engine, a sound engine, a video encoder, a USB1.1 device controller, a four-channel DAC/ADC, and other peripherals including four DMAs, a memory controller, four UARTs, an I2S, a key-pad controller, an interrupt controller, a two-channel watchdog timer, a PWM, and a GPIO.

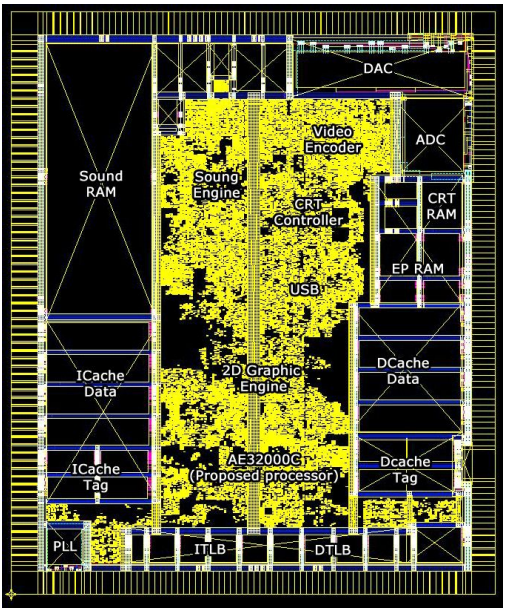


Fig. 4. Layout of the SoC equipped with the proposed DSP-enhanced EISC processor

5 Conclusions

In this paper, we have introduced a DSP-enhanced embedded microprocessor based on the EISC architecture. In order to accelerate DSP application with as little extra hardware as possible, we propose various enhancement schemes: some schemes exploit the special features of the EISC, including the `leri` instruction; and some schemes are to overcome the inherent deficits of the EISC, including the insufficiency of the instruction bits and the insufficiency of GPRs.

We adopt the SIMD architecture and tailor it to reduce the hardware complexity and the packing overhead. To improve the performance of SIMD architecture, we propose a loop-efficient address generation unit. The proposed address generation unit is designed to support commonly used memory addressing modes in DSP applications with low hardware complexity. We also adopt a register extension scheme to reduce performance degradation due to the register spilling.

The proposed DSP-enhanced processor has been modeled in Verilog HDL and synthesized using a 0.18μm CMOS standard library. The proposed DSP enhancements cost about 10300 gates and not increase the clock frequency. Our simulations and experiments show that the proposed DSP-enhanced processor reduces the execution time of the DSP kernels considered in this work by 47.8% and the DSP applications by 29.3%. The proposed processor has been embedded in an SoC for video processing and proven in silicon.

Acknowledgements

The authors wish to acknowledge the CAD tool support of IDEC (IC Design Education Center), Korea and the financial support of Advanced Digital Chips Inc., Korea. The authors would also like to thank the anonymous reviewers for their valuable comments.

References

1. Francis, H.: ARM DSP-Enhanced Instructions White Paper, http://arm.com/pdfs/ARM-DSP.pdf
2. MIPS Tech. Inc.: Architecture Set Extension, http://www.mips.com/content/Documentation/MIPSDocumentation/ProcessorArchitecture/doclibrary
3. Cho, K.Y.: A Study on Extendable Instruction Set Computer 32 bit Microprocessor, J. Inst. of Electronics Engineers of Korea, **36-D(55)** (1999) 11–20
4. Lee, H., Beckett, P., Appelbe, B.: High-Performance Extendable Instruction Set Computing, Proc. of 6th ACSAC-2001 (2001) 89–94
5. Kim, H.-G., Jung, D.-Y., Jung, H.-S., Choi, Y.-M., Han, J.-S., Min, B.-G., Oh, H.-C.: AE32000B: A Fully Synthesizable 32-bit Embedded Microprocessor Core, ETRI Journal **25(5)** (2003) 337–344
6. Lee, C., Potkonjak, M., Mangione-Smith. H.: MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems, MICRO-30(1997) 330–335
7. ARM Ltd.: The Thumb Architecture Extension, http://www.arm.com/products/CPUs/archi-thumb.html
8. Kissell, K.D.: MIPS16: High-density MIPS for the Embedded Market, Technical Report, Silicon Graphics MIPS Group (1997).
9. Park, G.-C., Ahn, S.-S., Kim, H.-G., Oh, H.-C.: Supports for Processing Media Data in Embedded Processors, Poster Presentation, HiPC2004 (2004).
10. Cho,K.Y., Lim, J.Y., Lee, G.T., Oh, H.-C., Kim, H.-G., Min, B.G., Lee, H.: Extended Instruction Word Folding Apparatus, U.S. Patent No.6,631,459, (2003)
11. Talla, D., John, L.K., Buger, D.: Bottlenecks in Multimedia Processing with SIMD Style Extensions and Architectural Enhancements, IEEE Tras. of Comp. **52(8)** (2003) 1015–1011
12. Hennessy, J.L, Patterson, D.A.: Computer Architecture; A Quantitative Approach 3rd Ed., Morgan Kaufmann Publishers (2003)
13. Jayaraj, J., Rajendran, P.L., Thirumoolam, T.: Shadow Register File Architecture: A Mechanism to Reduce Context Switch Latency, HPCA-8 (2002) Poster Presentation
14. Samsung Electronics: STD130 0.18um 1.8V CMOS Standard Cell Library for Pure Logic Products Data Book, Samsung Electronics (2001)
15. Advanced Digital Chips Inc.: GMX1000: A High Performance Multimedia Processor User Manual, Advanced Digital Chips Inc. (2005)

An Intelligent Sensor for Fingerprint Recognition

Salvatore Vitabile[2,3], Vincenzo Conti[1],
Giuseppe Lentini[1], and Filippo Sorbello[1,3]

[1] Dipartimento di Ingegneria Informatica, Universita' di Palermo,
Viale delle Scienze, Edificio 6, 90128, Palermo, Italy
{conti, sorbello}@unipa.it, lentini@csai.unipa.it
[2] Dipartimento di Biotecnologie Mediche e Medicina Legale, Universita' di Palermo,
via del Vespro, 90127, Palermo, Italy
vitabile@unipa.it
[3] Istituto di CAlcolo e Reti ad alte prestazioni, Italian National Research Council,
Viale delle Scienze, Edificio 11, 90128, Palermo, Italy

Abstract. In this paper an intelligent sensor for fingerprint recognition is proposed. The sensor has the objective to overcome some limits of the fingerprint recognition software systems, as elaboration time and security issues related to fingerprint transmission between sensor and processing unit. Intelligent sensor has been prototyped using the Hamster Secugen sensor for image acquisition and the Celoxica RC1000 board, employing a Xilinx VirtexE2000 FPGA, for image processing and analysis. Resources used, elaboration time as well the recognition rates in both verification and identification modes are reported in the paper. To the best of our knowledge, this is the first implementation for a full hardware implemented fingerprint recognition system.

1 Introduction

Biometric based systems for personal identification are always an open research issue. In literature many approaches have been proposed to develop fingerprint recognition systems. Generally, they are characterized by three main steps: image acquisition, 'biometric signature' extraction, matching between the acquired biological signature and the stored one.

Fingerprint minutiae extraction task is a very critical and complex step, so, different dedicated software algorithms have been proposed in literature [1], [2], [4], [5], [6], [7], [8], [9].

In this paper an intelligent hardware sensor for fingerprint recognition is proposed. Sensor prototype has been developed using the Celoxica RC1000 board [12]. The board employs a 2M gates Xilinx VirtexE FPGA [13]. The sensor implements ad hoc image processing algorithms selected evaluating both their performance when implemented in fixed point arithmetic and their requested hardware resources.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 27–36, 2005.
© IFIP International Federation for Information Processing 2005

The proposed intelligent sensor is composed by a *Sensor Acquisition Module* (SAM) and a *Sensor Processing Module* (SPM). The first one is based on Hamster Secugen sensor [19] for fingerprint image acquisition. The second one is an FPGA based prototype implementing the whole fingerprint recognition chain. Modules have been installed on a standard workstation and their communication exploits standard PCI bus.

The proposed system has been tested using 384 fingerprints belonging to 96 different people, and the F.A.R. (False Acception Rate) and the F.R.R (False Rejection Rate) have been used to verify its performance. Experimental trials shows that an interesting working point could be reached by the system with a F.A.R. of about 1% with the related F.R.R. of 8%.

The proposed system can be employed as an automatic fingerprint discriminator, too. The system has been evaluated with an identification test where each fingerprint has been compared with each database item in order to find a similarity index. The obtained results show that the processed image is in the subset composed by the 5 most similar fingerprints with a percentage of 84%.

The paper is organized as follow. Some related works are briefly described in section 2, whilst in section 3 some guidelines for algorithms profiling in terms of execution time and FPGA resources are presented. The proposed system as well as each processing phase are described in section 4. In section 5, both system elaboration times and recognition rates are presented. Finally, in section 6 the conclusion of this work is reported.

2 Related Works

In literature many approaches have been proposed and many software systems have been implemented to develop fingerprint based recognition system [1], [2], [5], [6], [8]. Generally, these systems exploit filters and image enhancement algorithms [21], classification algorithms and matching techniques and they are developed with standard high level programming languages on general purpose computers.

In [10] an hardware fingerprint recognition system is presented. However, in the system the fingerprint matching phase has not been developed. The rest of the fingerprint processing tasks were implemented in a FPGA device with a clock frequency of 27,65 Mhz, and a processing time of 589,46 ms.

3 The Hardware Design Guidelines

FPGA devices are widely used for rapid system prototyping. However, an efficient image processing algorithms implementation on FPGA requires an algorithms profiling phase before their implementation. Fingerprint processing algorithms have been analyzed and evaluated in order to optimize FPGA used resources, system elaboration time and result accuracy.

With the respect the FPGA requested resources, each image processing algorithm must be evaluated through the number of loop, the presence of recursion,

the number of divisions, square roots operators, and powers different from 2, the presence and the dimension of typical high level language structures as "union", "array", and "circular lists" [3]. Further analysis on image processing algorithms concerns their inclination for parallel and/or pipeline implementation.

The above points are very critical for an high efficient system implementation and they concern each phase of the identification system. The Quantitative Index, *QI*, to individualize the *a priori* performance of the fingerprint processing algorithms for the embedded solution is now introduced:

$$QI = X + 0.6 * Y + 0.3 * Z + 0.05 * W \tag{1}$$

where X is the number of the loops in the algorithm, Y is the number of recursion and dynamic structures in the algorithm, Z is the number of division and square root operators and powers different from 2 in the algorithm, and W is the number of union and array data structure in the algorithm.

For each phase of the fingerprint processing tasks, several algorithms have been profiled, modified and re-profiled in order to optimize system performance with respect to the hardware resources, processing time and recognition rate.

4 The Intelligent Sensor

As pointed out before, most of the proposed solutions are developed with standard high level programming languages on general purpose computers. In this paper the authors present an intelligent sensor that is able to acquire a fingerprint image, process it and select the corresponding database item for person identification. The sensor is composed by a *Sensor Acquisition Module* (SAM) and a *Sensor Processing Module* (SPM). The first one is based on Hamster Secugen sensor [19] for fingerprint image acquisition. The second one is a FPGA based prototype implementing the whole fingerprint recognition chain.

With more details the SPM is based on five sequential phases: the normalization phase; the binarization phase; the thinning phase; the minutiae extraction phase and the matching phase. In Figure 1 the SAM, the SPM and their relative connections with the Host buses are depicted. The SAM, based on Hamster Secugen sensor [19], acquires a fingerprint image. Successively the SAM transfers the acquired image to the SPM using both the host expansion bus and the host PCI bus. The SPM prototype has been developed on the RC1000 Celoxica board [12] equipped with a 2M gates Xilinx VirtexE FPGA [13]. SPM communications use only the host PCI bus and its clock has been set to 90/4 MHz in order to guarantee the correct data exchange between FPGA and board RAM. The RAM is used to store the fingerprint database, i.e. fingerprint image coding.

Exploiting the high data parallelism of the application, different algorithms have been parallelized. In addition, fingerprint processing phases have been pipelined in order to increase execution time as well as the final throughput. In what follows, the FPGA implementation of the five sequential phases for fingerprint recognition will be described.

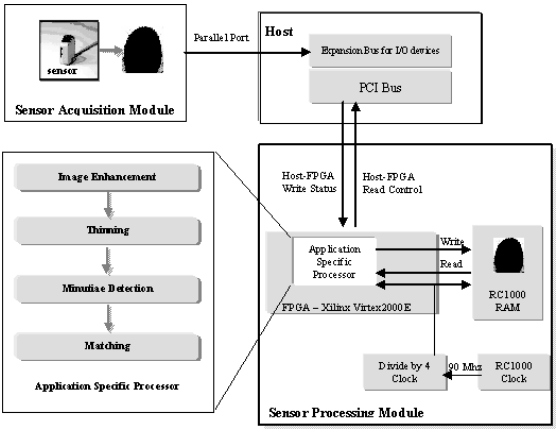


Fig. 1. In the figure is depicted the Sensor acquisition Module, the Sensor Processing Module and their communication with Host and Celoxica board

4.1 Fingerprint Normalization on FPGA

In this phase the undesirable fingerprint faults are reduced, because they can produce analysis mistakes [15]. The sensor fingerprint images could have low quality due to the non-uniform contact between user finger and sensor. Consequently, an adaptive normalization algorithm [14] based on local property of the fingerprint image considered is adapted for its efficient digitalization.

The sensor fingerprint image has been sub-sampled to reduce processing time eliminating the redundant information about the thickness of the ridges. So, the 300X260 pixels sensor image have been sub-sampled obtaining a 150x260 pixels image. Exploiting data parallelism of the adaptive normalization algorithm [14], each sensor image have been divided in four 75X130 sub-images for parallel processing, since the RC1000 board is equipped with four RAM memory banks. Sensor images are codified with 256 grey levels. We use 8 bit for each pixel storing four pixels in each RAM cell (32 bit).

The adaptive normalization algorithm [14] is based on four parameters: M_0, VAR_0, M and VAR. Experimental trials conducted on our sensor images show that they range around fixed values: 100, 250, 38, 5190, respectively. Fixing the above values, the equations to calculate the normalized pixel becomes more simple without the use of the square root and division operators:

$$G(i,j) = M_0 + (I(i,j) - M) << 2 \qquad if \qquad I(i,j) \geq M \tag{2}$$

$$G(i,j) = M_0 - (I(i,j) - M) << 2 \qquad if \qquad I(i,j) < M \tag{3}$$

where I(i,j) is the intensity of the pixel at the i_{th} row and j_{th} column and G(i,j) the relative normalized value.

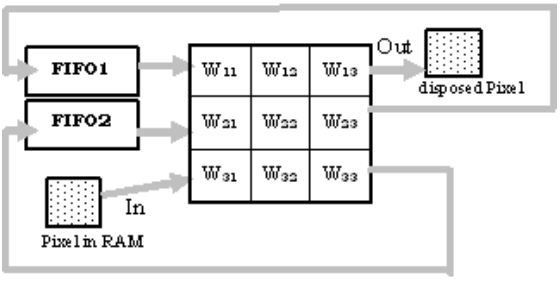


Fig. 2. An example of a pipelined 3x3 moving window implemented to perform the image binarization process

4.2 Fingerprint Binarization on FPGA

The main problem of this phase is a good threshold value selection for image binarization, because a wrong threshold value could erase real minutiae in the fingerprint image. In our solution two algorithms are used to improve the binarization step: the median filter algorithm [14], and the iterative thresholding algorithm [15].

The median filter is a non-linear digital filter that is able to preserve sharp signal and to remove impulse noise. With respect iterative thresholding algorithm, the analysis reported in [1], [2], shows that the best results are obtained using two thresholds S1 and S2. The thresholding operation is applied using an iterative method with a moving 3x3 kernel that find a solution for pixels whose intensity is greater than S1 and less than S2, with S1¡S2.

In the FPGA algorithms implementation, the main problem is the median filter implementation for its high computational cost. A 7x7 moving window was used to implement the median filter in the FPGA. The moving window was stored in a FIFO buffer and FPGA execution was pipelined to improve performance.

Successively, the median filter output pixels were processed by the thresholding operation. Experimental trials had shown that S1=140 and S2=170 give good binarization results. Following the median filter approach, a 3x3 moving window was implemented to complete the image binarization process.

In Figure 2 the pipelined 3x3 moving window elaboration is depicted: at each time the processed pixel is in the position (2,2). However each pixel starts its elaboration from the position (3,1) and two FIFO buffers are used to move it among the rows until the position (1,3).

4.3 Fingerprint Thinning on FPGA

Most of recent thinning algorithms were designed to optimize the execution time since they usually require a very high processing time. The techniques proposed in literature are based on morphological algorithms, sequential algorithms, parallel algorithms. Morphological thinning algorithms present good elaboration

time but too many resources are requested for their implementation. Sequential thinning algorithms show bad elaboration performances. This limit is overcome by the parallel thinning algorithms.

Following the proposed guidelines, the MB2 algorithm [20] and the Zhang-Suen algorithm [16] were profiled. The first algorithm uses the recursive operation, with an high resources consumption. The second algorithm is based on moving kernels without any recursion. In addition, the algorithm can be parallelized since for a faster execution.

In the FPGA implementation, the binary image was split in four sub-images (we exploit the RC1000 memory banks). Sub-images thinning based on moving kernels was pipelined as shown for the binarization phase.

4.4 Fingerprint Minutiae Extraction on FPGA

The minutiae, i.e. ridge endings and ridge bifurcations, are usually detected in the thinned fingerprint image [11]. Due to the presence of either original noise or preprocessing caused noise, the thinned image contains a large number of false minutiae.

Among the profiled algorithms, the Tico and Kuosmanen one [17] extracts every true minutia in one step without any noise reduction step. The same authors in [18] present a new version of their algorithm to remove the false minutiae caused by broken ridges into a 9x9 window.

The Tico and Kuosmanen algorithm works on adjacent matrixes of pixels so it was immediately pipelined in the related FPGA implementation. Algorithm performance were improved adding the false minutiae elimination in the boundary of the image. However, the new version of the Tico and Kuosmanen algorithm leaves an high number of false minutiae caused by broken ridges. Using the direction of every ridge-ending, the number of the erased false minutiae increases significantly.

4.5 Fingerprint Matching on FPGA

Several matching algorithms proposed in literature have been profiled. Algorithms based on pattern matching give the best performance due their possibility of parallel implementation. With more details, the algorithms proposed in [1], [2], were adapted for their efficient digitalization.

A fingerprint minutiae descriptor is used to implement the pattern matching. With more details, a list of both minutiae spatial coordinates and minutiae direction are used as descriptor to codify the fingerprint image. The algorithm receives as input two descriptors extracted from the enrolled fingerprint image and from the on-line processed image. As result, the algorithm gives a matching score with the similarity degree of the two fingerprints. A threshold is applied to the above matching score to decide if the processed fingerprints belong to the same person.

5 Experimental Results

5.1 Hardware and Software Environments

The proposed intelligent sensor is composed by a *Sensor Acquisition Module* and a *Sensor Processing Module*. The first one is based on Hamster Secugen sensor [19] for fingerprint image acquisition. The second one is an FPGA based prototype implementing the whole fingerprint recognition chain. Modules have been installed on a standard workstation and their communication exploits standard PCI bus. Sensor prototype has been developed using the Celoxica RC1000 board [12]. The board employs a 2M gates Xilinx VirtexE FPGA [13]. FPGA programming has been performed using both Celoxica DK2 [12] and Xilinx ISE [13] development environments. Algorithms have been described using the an algorithmic-like hardware programming language: the Handel-C language [12].

5.2 Elaboration Time

The board work frequency is limited by the work frequency of the RAM memory banks (see Figure 1). Celoxica RC1000 board manual suggests 25 MHz as maximum work frequency for board memory. We have performed several reading/writing data tests with the board memory banks. A clock frequency of 90/4 MHz, with 90 MHz the main board frequency, has assured proper and correct memory operations.

In the Table 1 some information about the implemented pipelined steps are illustrated in detail. With more details, the processed pixels mask dimension, the latency and running clock cycles, the real board work frequency and the obtained execution time are reported for each of the 5 processing steps.

Table 1. Processed mask dimension, latency and running clock cycles, real board work frequency and obtained execution time for each of the 5 processing steps

Processing Steps	Latency Clock Cycles	Running Clock Cycles	Working Freq.(MHz)	Execution Time(ms)
Normalization	8	6	20.9	12.5
Binarization	60	10	20.9	18.6
Thinning	83	73	21.1	39.8
Minutiae Detection	95	13	20.6	24.3
Matching	23	21	20.5	6.75

5.3 Resources Analysis

As pointed out before, both Celoxica DK2 [12] and Xilinx ISE [13] development environments have been used for FPGA programming. Algorithms have been described using the Handel-C language [12]. The Map Report tool inside the Xilinx ISE [13] development environment gives an output with the FPGA resources used by the fingerprint elaboration chain. Table 2 summaries the used resources for the 2M gates Xilinx VirtexE FPGA [13].

Table 2. FPGA resources used by the fingerprint elaboration chain

Resources Type	FPGA Total Resources	Used Resources	Used Resources (%)
GCLK	4	2	50
GCLKIOB	4	1	25
IOB	404	273	67
SLICE	19200	16178	84
LUT	38400	27027	70
BLOCKRAM	160	6	3

5.4 Recognition Rates

In the registration phase, a sequence of 4 fingerprints of the same finger was acquired in different days. So for each person 4 biological signatures or descriptors were extracted and stored in the 4 board memory banks.

In the matching phase, the on-line extracted biological signature is compared with the registered signatures for a typical verification. With more details, the proposed sensor has been evaluated in both verification tasks (a username is used to select the database item and perform a 1-¿1 match) and identification tasks (no username is used so 1-¿all matches must be performed in order to select the highest matching score candidate).

Due the sensor design features, four security levels could be chosen on the basis of the matched biological signature number: the *basic security level*, if the on-line extracted biological signature matches one of the four stored signatures; *normal security level*, if the on-line extracted biological signature matches two of the four stored signatures; *high security level*, if the on-line extracted biological signature matches three of the four stored signatures; *very high security level*, if the on-line extracted biological signature matches each of the stored signatures. Exploiting both Celoxica RC1000 memory banks and the *par* feature of the Handel-C language, the four fingerprint matches are parallel executed. Experimental trials have demonstrated that the *normal security level* is the best trade-off between security and performance.

Intelligent sensor recognition performance were evaluated using the following indexes: F.A.R. (False Acception Rate) and the F.R.R (False Rejection Rate). Experimental tests were executed on a sample of 384 fingerprints taken by the Secugen sensor [19] belonging to 96 different individuals.

The first set of experiments was conducted to evaluate the performance of the proposed intelligent sensor in verification mode. So with a matching score of 95%, intelligent sensor performance are summarizes by FAR=1,07% and FRR=8,33%. In Figure 3 are plotted FAR and FRR vs the matching score for the normal security level. With the second set of experiments we address the problem of individual identification in large fingerprint database. Experiments aim with a database subset selection, composed by only 5 fingerprints, that always contains the processed fingerprint. In the best case, a recognition rate of 84% with a maching score of 93% is obtained, i.e. the processed fingerprint is among the 5 selected ones.

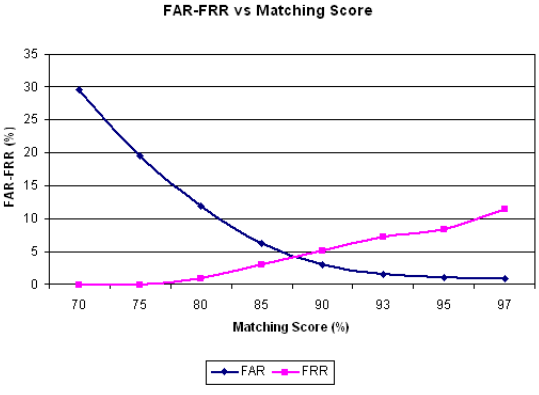


Fig. 3. FAR and FRR vs the matching score for the normal security level

6 Conclusion

In this paper an hardware intelligent sensor for fingerprint recognition is proposed. The sensor has been tested with 384 fingerprints belonging to 96 different people. Sensor performance have been evaluated with F.A.R. and F.R.R indexes in both verification and identification modes. In the verification mode, a F.A.R. of about 1% and a F.R.R. of about 8% are obtained. The sensor has been used as fingerprint discriminator, too. Each processed fingerprint is among the set of 5 similar fingerprints with recognition rate of 84%. The obtained experimental results are comparable with the full software recognition systems. In Bonato at al. [10] an incomplete hardware implemented recognition system (the matching process was not implemented) was proposed. System elaboration time was about 590 ms, while the proposed sensor elaboration time is about 102 ms, considering the full fingerprint matching process.

References

1. V. Conti, G. Pilato, S. Vitabile, F. Sorbello, *A Robust System for Fingerprints Identification*, Knowledge-Based Intelligent Information Engineering System and Allied Technologies, Crema September 2002, pp. 1162-1166
2. V. Conti, G. Pilato, S. Vitabile, F. Sorbello, *Verification of Ink-on-paper Fingerprints by Using Image Processing Techniques and a New Matching Operator*, AI*IA September 2002, pp. 594-601
3. S. Vitabile, A. Gentile, S.M. Siniscalchi, F. Sorbello, *Efficient Rapid Prototyping of Image and Video Processing Algorithms*, Proc. of EUROMICRO Symposium on Digital System Design - Architectures, Methods and Tools, Rennes 2004, pp. 452-457, IEEE Computer Society Press.
4. Jain A., *A Multichannel Approach to FingerPrints Classification*, IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol.21, n.4, 1999, pp. 348-358

5. Jain A., *On-Line Fingerprint Verification*, IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol.19, n.4, 1997, pp. 302-314
6. H. Lin, *Fingerprint Image Enhancement*, IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol.20, n.8, 1998, pp. 777-789
7. Prabhakar S., Jain A., W. Jianguo, *Minutiae Verification and Classification*, Department of Computer Engineering and Science, University of Michigan State, East Lansing 1998
8. Miklos, Zsolt, Vajna, Kovacs, *A Fingerprint Verification System based on Triangular Matching and Dynamic Time Warping*, IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol.22, n.11, 2000, pp. 1266-1276
9. Cappelli R., Lumini A., Mario D., Maltoni D., *Fingerprint Classification by Directional Image Partitioning*, IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol.21, n.5, 1999, pp. 402-421
10. V. Bonato, R.F. Molz, J.C. Furtado, M.F. Ferro, F.G. Moraes, *Propose of a hardware implementation for fingerprint systems*, UNISC - Departamento de Informatica Santa Cruz-Brazil, PUCRS - Faculdade de Informatica porto Alegre - Brazil
11. D. Maio, D. Maltoni, *Direct Gray-Scale Minutiae Detection in Fingerprints*, IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol.19, n.1, 1997
12. Celoxica Ltd.[on line], http:// www.celoxica.com
13. Xilinx Inc. [online], http:// www.xilinx.com
14. L. Hong, Y. Wan, A. Jain, *Fingerprint Image Enhancement: Algorithm And Performance Evaluation*, IEEE Transactions On Pattern Analysis And Machine Intelligence, 1998, Vol.20, N8, pp. 777-789
15. I. Emiroglu, M.B. Akhan, *Pre-Processing of FingerPrint Images*, European Conference on Security and detection, Conference Publication n.437, IEEE 1997
16. T.Y. Zhang, C.Y. Suen, *A fast parallel algorithm for thinning digital patterns*, Comm. ACM., 27(3) pp. 236-239, 1984
17. M. Tico, P. Kuosmanen, *An Algorithm for Fingerprint Image Postprocessing*, IEEE Transactions On Pattern Analysis And Machine Intelligence, pp.1735-1739, 2000
18. M. Tico, P. Kuosmanen, *Fingerprint Matching using an orientation-based Minutia Descriptor*,IEEE Transactions On Pattern Analysis And Machine Intelligence, Vol.25, no.8, 2003
19. *SecuGen FDx Developer's Gui*, Copyright 1998-2001 SecuGen Corporation and NITGen Co., Ltd. http://www.secugen.com/
20. T.M. Bernard, A. Manzanera, *Improved Low Complexity Fully Parallel Thinning Algorithm*, International Conference on Image Analysis and Processing, pp. 215-220, Venice, Italy, September 1999
21. V. Conti, G. Milici, G. Vetrano, S. Vitabile, F. Sorbello, *Fingerprint Registration Using Specialized Genetic Algorithms*, accepted in the 8th International IEEE EUROCON Conference, Belgrade, November 21-24, 2005

Exploiting Register-Usage for Saving Register-File Energy in Embedded Processors

Wann-Yun Shieh* and Chien-Chen Chen

Department of Computer Science and Information Engineering,
Chang Gung University, Taiwan
259 Wen-Hwa 1st Road, Kwei-Shan Tao-Yuan, 333, Taiwan
TEL: 886-3-2118800-3336, FAX: 886-3-2118700
wyshieh@mail.cgu.edu.tw

Abstract. Low power register file design plays an important role in an embedded processor. In this paper, we exploit register-usage in a program to find out unused registers, and turn these unused registers into low power mode by annotating power-controlling instructions. The whole work is performed by applying the hardware/software co-design principle. For the hardware part, we propose a voltage-scaling control logic to supply voltages for each register. For the software part, we propose a power-controlling-code annotation approach to determine the voltage scaling behavior for each register. Simulation results show that the proposed approach outperforms the other related approaches in terms of the energy-delay product.

1 Introduction

In recent years, low power register file researches are lacking even though many other low-power hardware components, like BTB, ROB, TLB, etc, have been proposed. In general, the energy consumption of register file in an embedded processor is quite large. When we explore the sources of major energy consumption in register file, we find that all registers in an embedded-processor always keep active mode (1Volt) during program execution. This is because the registers must retain the values, which are produced from computing process. However, not all registers need to stay in active mode. The registers that need not to stay in active mode are called unused registers. Unused registers have two types. The first type is that the register values are temporarily not to be used, and the second type is that the register values are no longer to be used. These unused registers become the major sources of wasted energy consumption during program execution. Hence, finding out these unused registers in a program, and turning these unused registers into low power mode, i.e., drowsy mode (0.3Volt) or destroy mode (0 Volt), will result in large energy saving.

The objective of paper is to exploit register-usage in a program to find out unused registers for register file, and turn these unused registers into low power mode without large performance penalty. In order to achieve this objective, there are three important issues:

* Corresponding author.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 37 – 46, 2005.
© IFIP International Federation for Information Processing 2005

Issue 1: How to analyze register-usage in a program to find out unused registers?
Issue 2: How to control register voltages by different working modes?
Issue 3: How to limit energy and performance penalty?

The rest of this paper is organized as follows. Section 2 introduces and discusses related works, such as the dynamic voltage scaling technology and the low-power register file design. Section 3 proposes hardware/software co-design approach to exploit register-usage in a program for register energy saving. Section 4 discusses evaluation environment and evaluation results, including the metrics of register file energy consumption, processor energy consumption, performance penalty and energy-delay product. Section 5 gives conclusion and future works.

2 Related Works

In this section, we discuss the development of the dynamic voltage scaling (DVS) technique, which is a well-known approach in reducing hardware energy consumption. Then we discuss some previous research about low-power register file design.

2.1 Dynamic Voltage Scaling Technique

The dynamic voltage scaling (DVS) technique is widely used to reduce power consumption or leakage energy for hardware components. The main idea of the DVS technique is to dynamically control supply voltages to a hardware by its working modes, including the active mode, the drowsy mode and the destroy mode, based on the access behaviors.

Most previous researches about the DVS technique focus on hardware solutions by utilizing hardware monitor to dynamically control the supply voltages. However, pure hardware solution has its drawbacks. For example, it may increase more hardware area due to the need of extra logic to control voltage scaling. In addition, the hardware solution does not precisely analyze overall program execution behavior, e.g., register usage; as a result, it cannot get much energy saving.

Due to these reasons, in recent years, the trend of the DVS technique has focused on coupling with the hardware/software co-design principle. This kind of approaches uses software analysis to get sufficient information from user programs, and then uses these information to help hardware components to switch among different working modes to get energy saving.

2.2 Hardware/Software Co-design for Register File

In [1], authors used the hardware/software co-design approach with the DVS technique to reduce register file energy. They analyze loop sections in a program to find unused registers, and then inserted power-controlling instructions in the loops to turn unused registers into drowsy mode. The advantages of this approach are easy analysis and simple implementation, because it focuses only on the loop sections. Also, it can reduce power-controlling instruction annotations, which may be the potential performance penalty. However, this approach, which did not consider other non-loop sections in a program, may possibly get little energy saving for the whole program.

3 Hardware/Software Co-design Approach

In this study, we propose a novel hardware/software co-design approach to exploit register usage for register-file energy saving. The proposed approach can be partitioned into the hardware part and the software part.

In the hardware part, we propose a voltage-scaling mechanism that contains two major components: the connected power-state-register design and the voltage-scaling control logic. The connected power-state-register is to record the power states (i.e., working modes) for multiple registers. The voltage-scaling control logic is to control different supply voltages to registers by individual working modes.

In the software part, we propose a power-controlling code annotation method that has two major components, including the power-controlling instruction design and the power-controlling code generation. The power-controlling instruction is to write a value into a power-state-register to change the working modes for multiple registers. The power-controlling code generation is to analyze the register-usage in a program and annotate power-controlling instructions into the program, based on a proposed voltage-scaling strategy, to determine the register working modes.

3.1 The Hardware Part: Voltage-Scaling Mechanism

Fig. 1 shows the voltage-scaling mechanism. When a power-controlling instruction writes a new value into a connected power-state-register, the voltage-scaling control logic will switch the supply voltages to corresponding registers.

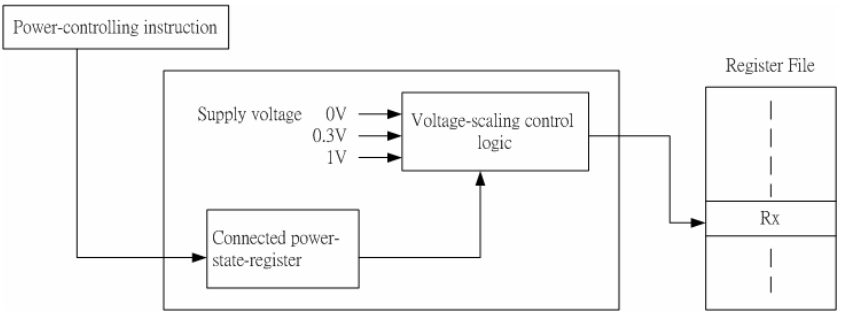


Fig. 1. Voltage-scaling mechanism

3.1.1 Connected Power-State-Register Design

The design goal of the proposed power-state-register is to reduce power-controlling instruction annotations for saving potential performance penalty. In conventional design, each register is controlled by an individual 1- or 2-bit power-state-register, causing that multiple sequential power-state-register updates require multiple sequential power-controlling instruction annotations. Therefore we "connected" those individual power-state-registers into a single multi-bit register, called the connected power-state-register, for merging multiple sequential updates into just one annotation. The number of the connected power-state-registers for a register file can be

determined by $\lceil 2n/m \rceil$, where n is the total register number to be controlled, and m is the default length of a connected power-state-register.

3.1.2 Voltage-Scaling Control Logic

The design goal of the voltage-scaling control logic is to control different supply voltages by different working modes, including the active mode (1Volt), the drowsy mode (0.3Volt), and the destroy mode (0Volt). In [1], authors proposed a control logic by applying the "switch" characteristic of transistors to turning on/off the supply voltages for each register. In this paper, we use the same control logic, but the power-state-register is replaced by the proposed connected power-state-register, as shown in Fig. 1.

3.2 The Software Part: Power-Controlling Code Annotation

3.2.1 Power-Controlling Instruction Design

The power-controlling instruction writes a value into the connected power-state-register to control working modes for multiple registers. Through the connected power-state-register design, each value-update (i.e., instruction annotation) requires only few ISA modifications. We can directly use the instructions, e.g., ADD or SUB, to change the value of a connected-power-state-register at the time that the working modes of some registers should change. These instructions will be annotated in the program execution code to control the working modes for each register.

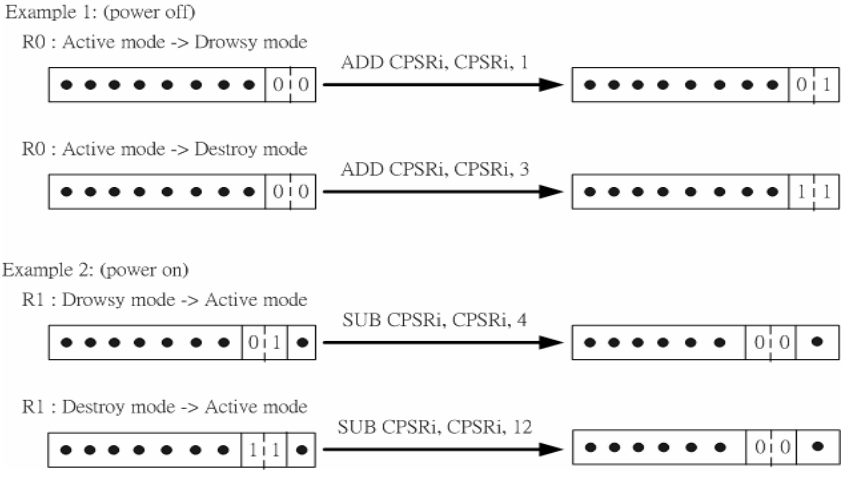


Fig. 2. Two examples of changing the working modes by ADD or SUB instructions

Fig. 2 shows two examples to change the working modes of registers R0 and R1 through ADD and SUB instructions, respectively. In example 1, if we want to change the working mode of R0 from the active mode to the drowsy mode, we can update the

value of the connected power-state-register (CPSRi) by annotating the instruction "ADD CPSRi, CPSRi, 1". This is because the bit-0 and bit-1 of the CPSRi control the supply voltages to R0, and state "01" represents 0.3-Volt drowsy mode. Similarly, we can annotate the instruction "ADD CPSRi, CPSRi, 3" to represent the change of working mode from the active mode (00) to the destroy mode (11). Example 2, on the other hand, shows how to change the working mode of R1 (i.e., bit-2 and bit-3) from the drowsy mode (01), or from the destroy mode (11) to the active mode (00).

3.2.2 Power-Controlling Code Generation

To precisely predict register-usages in a program, we take the trace file for analysis. The reason to do so is that we can exactly measure how the power-controlling code affects power consumption and performance penalty during program execution. First, the program is partitioned into sections for register-usage analysis. In [1] authors take the loop body as analyzed sections (we call it the Loop-based approach for convenience). In this paper, we propose another two more straightforward partitioning approaches, i.e., partition the program by fixed interval (called the FI-based approach), and by basic blocks (called the BB-based approach), for sections not limited to loops. Though these three approaches (FI-, BB-, and Loop-based) are employed for simple analysis and easy implementation, they cannot precisely reflect register-usage. A more "making-sense" option would be the approach that partitions the program by register-live-ranges (we call it the RLR approach).

Fig. 3 shows the register-live-range graph for a piece of instruction sequences in program basicmath of benchmark MiBench [3]. (In Fig. 3, we draw the live-ranges by solid lines, as well as the access point of register contents by small circles.) From the register-live-range graph, we found that the distribution of live-ranges is not so uniform; there did exist "sparse" and "dense" sections. The sparse section, in fact, represents the section that has more chances we can get energy saving from it, due to large amounts of unused registers. On the other hand, the dense section represents the section that the register pressure is heavy, and it is not worth to annotate extra power-controlling instructions to enlarge either energy or performance penalties. Therefore, in the RLR approach, only sparse sections will be taken for register-usage analysis. The partitions of the sparse sections and the dense sections can be performed by setting a scanning line across the register-live-ranges. When the scanning line scans along each instruction, if intersection numbers is less than a threshold (IT), we call it in a sparse section, and vice versa. After scanning the whole program, we get the sparse and dense sections alternatively appeared in the register-live-range graph.

In a sparse section, there are two types of ranges can be considered for energy saving. The first type, called the type-I range as shown in Fig. 3, means that the corresponding register is temporarily unused, and we may change it to the drowsy mode (the data will be kept for later use). The other type, called the type-II range, means that the value in the corresponding register is no longer to be used, and we can change it to the destroy mode. For both types of ranges, we will apply the same energy cost function to judge whether it is worth to insert power-controlling instructions.

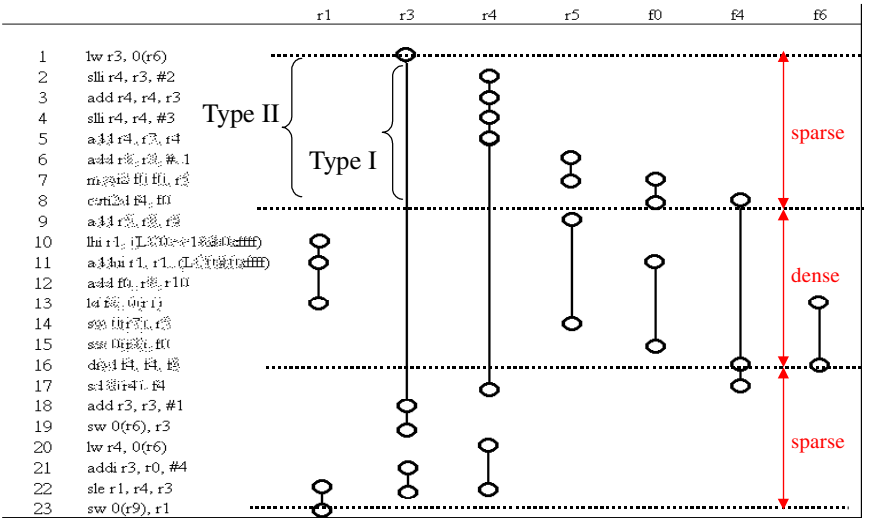


Fig. 3. Example of register-live-ranges. The solid lines represent the live ranges, and a small circle represents the access point for a value.

The RLR approach performs well balance between energy-consumption saving and performance penalty. First, by focusing only on the major sources of energy-consumption in sparse sections (i.e., type-I and type-II ranges), we can earn larger energy saving, but suffer less performance penalty. It is because only the sparse sections require power-controlling instruction annotations, not the whole file. Second, varying the IT threshold makes the size of sparse sections change differently; smaller IT results in more small-size sparse sections (and vice versa). These small-size sparse sections may not pass the energy cost function evaluation, and result in few instruction annotations. Thus, they earn less energy saving, but also suffer less performance penalty. (On the other hand, larger IT derives the opposite results.) The choice of IT, in fact, can be evaluated by experiments. Finally, by applying energy cost function evaluation to each instruction annotation, we can guarantee that any energy saving for register file would not cause extra energy consumption in pipeline; that is, minimize the side-effects for the whole processor energy.

We apply an energy cost function to each partitioned section to judge whether it is worth to annotate power-controlling instructions. Though the registers can achieve energy savings through working mode changes, the power-controlling instructions annotated may cause extra energy consumption in pipeline. Therefore, an energy cost function is used here to measure if energy saving of changing register working-mode is greater than the energy penalty of annotating extra instructions. If yes, then a voltage-scaling strategy will be activated to determine which working mode should a register change to. If no, nothing happens in this section, and the next partitioned section will be taken for analysis through the same processes.

4 Evaluation

In this section, we first present the evaluation configuration, including environment, parameters, and evaluation flow. Then we show evaluation results, in the terms of the register-file energy-consumption, the processor energy-consumption, the performance penalty, and the energy-delay product. In addition, we show how to select suitable IT threshold for the RLR approach.

4.1 Evaluation Environment

Table 1 shows evaluation environment, including target processor, benchmark, and approaches for comparison.

First, we use ARMulator to produce the trace files of benchmarks. Then the trace files are fed into the power-controlling-code annotation process. In this process, we would evaluate four approaches including FI, BB, Loop, and RLR –based approaches by energy and performance analysis.

Table 1. Evaluation environment

Simulator	Simplescalar 4.0 [9]
Energy model	Wattch 1.2 [4]
Target processor	ARM with 5 pipeline stages [10]
Benchmark	MiBench [3]
Approaches for comparison	FI, BB, Loop, RLR

4.2 Evaluated Results

A. Flexibility of the RLR approach – the effect of the *IT* threshold

The IT plays the role of balancing the trade-off between energy saving and performance penalty. Different IT values would result in different influences. Applying a large IT in the RLR approach makes more large-sparse-sections; that is, suitable for energy-oriented applications. On the other hand, applying a small IT in the RLR approach makes less sparse-sections; that is, suitable for performance-oriented applications. Fig. 4 shows that, for example, when we set the value of IT to 8, we get larger energy saving and less performance penalty. We can derive the similar results from the experiments for other benchmark programs in MiBench.

B. Register File Energy Consumption

Fig. 5 shows the register-file energy-consumption for the four approaches. In the FI-based approach, we set the fixed-interval (i.e., *n*) by average length of basic blocks. In the BB-based approach, we partition the programs by basic blocks. In the loop-based approach, we partition the programs by loops. And, in the RLR-based approach, we partition the programs by register-live- ranges, and set the value of IT to eight.

Fig. 5 shows that for register-file energy-consumption, the RLR approach averagely outperforms the FI, BB, and Loop approaches by 21%, 18%, and 28%, respectively. These improvements come from that the RLR approach considers not only the type-I "temporarily-unused" register-live-ranges, but also the type-II "no-longer-used" register-live-ranges, and the type-II situations can be proved as the major wasted energy sources.

C. Performance Penalty

Fig. 6 shows the performance penalty for the four approaches. In Fig. 6, on average, the loop-based approach outperforms the other approaches by 17% ~ 23%. This is because the loop-based approach only considers the loop-sections in a program; hence, it results in the fewest power-controlling code annotations. On the other hand, the RLR approach performs better than the FI and BB -based approaches. This is because it focuses only on the sparse sections instead of the whole program. Hence, for the RLR approach, the number of power-controlling-code annotations can be limited.

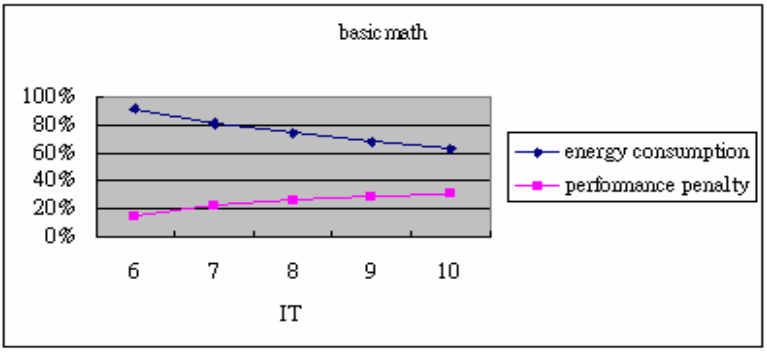


Fig. 4. Effect of IT in the RLR approach for basicmath of MiBench

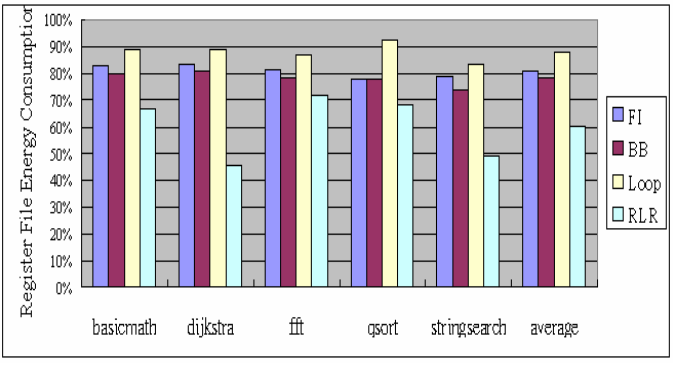


Fig. 5. Register-file energy-consumption for the four approaches

D. Energy-Delay Product

The energy-delay product is a widely-used metric to balance benefits in energy saving with any potential degradation in performance [2]. Fig. 7 shows the energy-delay products for the four approaches, and each approach is normalized to the FI approach. In Fig. 7, we find that the RLR approach would be a good choice among the others when the value of *IT* equals to 8. (It outperforms the other approaches by 21% - 67%.) This shows that the RLR approach gets a great balance between energy saving and performance penalty.

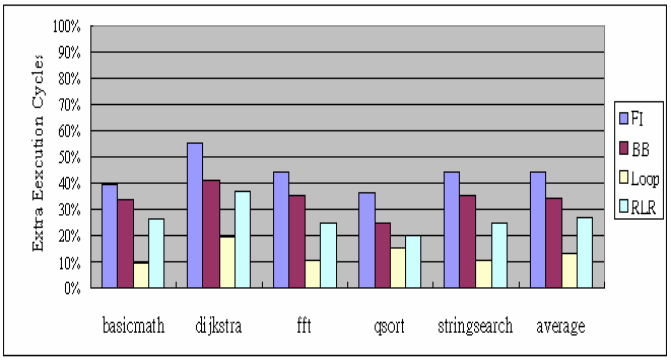


Fig. 6. Performance penalty for the four approaches

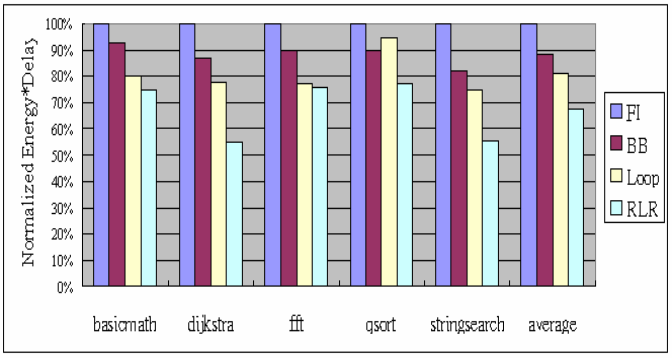


Fig. 7. Normalized energy-delay product for the four approaches

5 Conclusion

In this paper, we propose a hardware/software co-design approach for register-file energy saving. On the hardware part, we propose a voltage-scaling mechanism, including the connected power-state-register design and the voltage-scaling control logic. On the software part, we proposed a power-controlling code annotation mechanism, including the power-controlling instruction design and the power-controlling code generation. We partition the programs into sections to analyze register-usage. Simulation results show that the RLR-based approach outperforms the other partitioning approaches in terms of the energy, and energy-delay product. In addition, by varying the *IT* threshold in the RLR-based approach, the trade-off between energy saving and performance penalty can be well balanced.

There are some future works in this paper. For example, energy and performance penalty due to power-controlling code annotations could be reduced further by hardware supports. We know that a “power-on” instruction is annotated before a register tends to be read or updated. In fact, we can catch this register-usage information in early pipeline stages, e.g., ID stage, without compiler notification. By

forwarding a signal from the ID stage to a voltage-scaling control logic, we can power up a register on time for later use. Detailed design has been under development.

References

1. J. L. Ayala,et al.: Power-Aware Compilation for Register File Energy Reduction. International Journal of Parallel Programming, Vol. 31, No. 6 (2003)
2. W.Zhang, et al.: Reducing Instruction Cache Energy Consumption Using a Compiler-based Strategy. ACM Transactions on Architecture and Code Optimization, Vol.1, No. 1 (2004)
3. M. R. Guthaus, et al.: MiBench: A free, commercially representative embedded benchmark suite. IEEE International Workshop on WWC-4, (2001) 3 – 14
4. D. Brooks, et al.: Wattch: A Framework for Architectural-Level Power Analysis and Optimizations. Proceedings of the 27th International Symposium on Computer Architecture (2000) 83 – 94
5. J. L. Ayala, et al.: Reducing Register File Energy Consumption using Compiler Support. Spanlish Ministry of Science and Technology under contract TIC 2000-0583-C02-02.
6. J. L. Ayala, et al.: Power-aware Register Renaming in High-Performance Processors using Compiler Support. Spanlish Ministry of Science and Technology under contract TIC 2003-0708.
7. J. L. Ayala, et al.: Energy-Efficient Register Renaming in High-Performance Processors. Spanlish Ministry of Science and Technology under contract TIC 2000-0583-C02-02.
8. J. Abella :On Reducing Register Pressure and Energy in Multiple-Banked Register Files. Proceedings of 21st International Conference on Computer Design (2003) 14 – 20
9. www.simplescalar.com/docs/simple_tutorial_v4.pdf
10. Http://www.arm.com/documentation/Software_Development_Tools/index.html

Hardware Concurrent Garbage Collection for Short-Lived Objects in Mobile Java Devices

Chi Hang Yau, Yi Yu Tan, Anthony S. Fong, and Wing Shing Yu

Department of Electronic Engineering, City University of Hong Kong,
Tat Chee Avenue, Hong Kong
Anthony.Fong@cityu.edu.hk

Abstract. jHISC is an object-oriented processor for embedded system aiming at accelerating Java execution by hardware approach. Garbage collection is one of the critical tasks in a Java Virtual Machine. In this paper, we have conduct a study of dynamic object allocation and garbage collection behavior of Java program based on SPECjvm 98 benchmark suite and MIDP applications for mobile phones. Life, size, and reference count distribution of Java objects are measured. We found most Java objects die very young, small in size and have small number reference counts. Reference counting object cache with hardware write barrier and object allocator is proposed to provide the hardware concurrent garbage collection for small size objects in jHISC. Hardware support on write barrier greatly reduces the overhead to perform the reference count update. The reference counting collector reclaims the memory occupied by object immediately after the object become garbage. The hardware allocator provides a constant time object allocation. From the investigation, over half of Java objects can be garbage collected by the object cache that makes it unnecessary for these objects to copy to the main memory.

1 Introduction

Java is a widely used programming technology for developing applications on different kinds of mobile devices. Fast product development, portability and secure environment are the advantages that Java offers for the mobile devices. Connected Limited Device Configuration (CLDC) and Mobile Information Device Profile (MIDP) provide the core application functionality required by mobile applications, in the form of a standardized Java runtime environment and a rich set of Java APIs. Sun Microsystems announce that there are over 250 million handsets already in people's hands which support Java, over 200 different Java handset models (globally) and around 77 network operators' deploying Java based Web services in worldwide.

Java KVM is the traditional JVM on mobile devices, simple mark-sweep algorithms are used to collect garbage objects. The mark-sweep collector traverses the heap from the root to determine the reachability, and then sweep the heap by collect the reclaimed memory into a freelist. The major weakness of the classic mark-sweep GC method is that all other threads within JVM have to be suspended while the garbage collector is running. The suspension time is directly proportional to the number of objects on the heap.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 47 – 56, 2005.
© IFIP International Federation for Information Processing 2005

jHISC processor is proposed by Fong [1][2], which applies object-oriented concept into the architecture of the processor. This paper introduces a hardware garbage collection method based on reference counting algorithm, for jHISC processor.

The remainder of the paper is organized as follows: Section 2 is an overview of the related research. Section 3 is a research of the object behavior. Section 4 describes the design of the proposed garbage collector. Evaluation and further study are on the last two sections.

2 Background

2.1 Reference Counting

Reference-counting is a traditional garbage collection algorithm. The concurrent nature makes it suitable to be used in real-time embedded system. The traditional method reference counting was first developed for Lisp [13]. It was widely used in SmallTalk-80 [14] and Perl [15]. The basic theory of the reference-counting algorithm is that each object has an associated count of the references (pointers) to it. When a variable is assigned a reference to that object, the reference-count is incremented by one. Similarly, whenever a reference pointing to an object is deleted, its reference-count is decremented by one. When the reference-count becomes zero, it implies that the object is not referenced by other objects and the executing process cannot reach it. The memory occupied by this object is garbage and can be reclaimed by the collector.

The space overhead to store the reference counts is the weaknesses of the reference counting method. Extra space in each object is needed to store the reference count. For the worst case, the reference count field would have to be large enough to hold the total number of object references in the memory space [4]. Whenever a reference to an object is made, copied or deleted, it is necessary to access memory to update the object's reference count field. Furthermore, computation to update reference count degrades mutator performance.

There are researches try to apply reference counting algorithm into Java Virtual Machine (JVM). In JVM architecture, references to an object are created form three areas, which are other form objects heap, Java stacks and pc registers. Deutsch and Bobrow noted that most of the overhead is on counter updates from the frequent updates of local references (in stack and registers) [12]. Deferred reference counting is suggested by Blackburn to ignore frequent pointers mutations to the stack which eliminates most of their reference counting load [8]. It thus postpones all reclamation until it periodically enumerates the stacks.

Cyclic objects cannot be garbage collected because of its cyclic structure. Most of the research work of reference counting is used in generational garbage collection. Bacon and Rajan developed a current and localized algorithm to solve this disadvantage to make RC unnecessary to work with other garbage collection algorithms [11]. The algorithm is capable of collecting garbage even in the presence of simultaneous mutation and never needs to perform a global search of the entire object heap. Local search is performing after a possible root is found. That algorithm combines the tracing and reference counting collection. It is only an incremental garbage collection algorithm because pause occurs when there is local search.

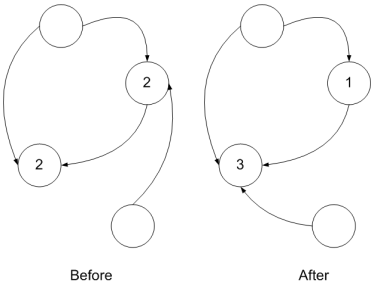


Fig. 1. Reference Count updates when a pointer is move

2.2 Garbage Collection in Mobile Java

Due to the popular of the Java enabled mobile phone, many researches are conducted to improve the performance of Java execution in the limited resource on mobile device. Active memory processor is introduced by Chang to reduce the overhead of memory management in Java [6]. A bitmap-based processor is used to work with standard DRAM cells. Memory availability and GC information are stored in a bitmap. There are Compete Binary Tree (CBT) and a bit-sweeper inside the processor to provide constant time garbage sweeping and object allocation. Limited reference field is supported in the processor. According to the study of Chang, it can reduce the amount of the memory usage by 77%.

Table 1. Description of MIDP Benchmark Applications

Applications	Description
WebViewer	Simple Web Browser
WormGame	Popular game on mobile photo
PhotoAlbum	Image viewer that can used to view image file like JPEG, GIF and PNG
AudioDemo	Audio player include in the MIDP reference implementation

The disadvantage of the idea of Chang is that the design of the hardware memory manager is complex. It increases the cost of the hardware and power consumption. The CBT design cannot utilize the memory space well and causes internal fragmentation in the heap.

3 Behavior of Java Objects

The Mobile Information Device Profile (MIDP) is a set of Java APIs which together with the Connected Limited Device Configuration (CLDC) to provide an environment for mobile devices, such as PDA and mobile phone. In this study, the object behavior of Java applications running on embedded devices based on MIDP 2.0 and CLDC 1.0.4 are investigated. There are no standard benchmark suites for MIDP applications available yet. Four applications are chosen form PennBench [3], which is a benchmark suite for embedded Java, defined by Java researchers.

In the MIDP study, we focus on the object life. We have modified the source code of the KVM to trigger garbage collector before every object allocation. The garbage collection scheme used on KVM is the simple Mark-Sweep algorithm. It is a Stop-the-World algorithm. Only garbage collector thread is active during garbage collection in progress, and other threads are suspended [5]. Object allocation is not allowed during garbage collection in progress. After the garbage collection is completed, no newly created object is in the object that makes sure all the memory used by the garbage objects are freed and available immediately for further object allocation.

Table 2. Accumulative Distribution of Object Size on Java MIDP applications

Applications	Percentage of objects which the size is smaller than <= 48 bytes
WebViewer	87.01%
WormGame	91.00%
PhotoAlbum	91.28%
AudioDemo	92.75%
Overall	90.51%

3.1 Size Distribution

The object size is the first category we are concerned. On KVM runtime environment, each object has an object header which is 2 words (a word is 32-bit) in size. The minimum size of an object in KVM is 8 bytes. Each data inside Java object would use a word to store, regardless if it is an integer or a character. In our study, we found that most object sizes are small. Over 90% of objects are equal to or smaller than 48 bytes (12 words).

3.2 Age Distribution

The relative age of an object we defined is the number of objects allocated between the object allocation and de-allocation events. According to the example of figure 2, there are two object allocation events between the allocation and de-allocation of object A. Based on the definition, the relative age of object A is 3. Applying the same rule on object C, the life of object C is zero.

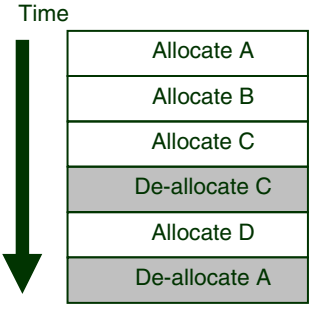


Fig. 2. Definition of relative age for Java objects

We found that more than half of the Java objects (~50%) have relative ages of 0, and over 90% of objects have relative ages less than 16. We can make a conclusion that the life span of a typical object in embedded Java environment is short. Half of the Java objects can be garbage collected immediately before an object allocation event.

Table 3. Accumulative Distribution of Object AGE on Java MIDP applications

Applications	AGE = 0	AGE <=15
WebViewer	30.92%	83.54%
WormGame	55.80%	92.41%
PhotoAlbum	62.95%	95.59%
AudioDemo	58.15%	92.81%
Overall	51.95%	91.09%

3.3 Reference-Count Distribution

Jikes RVM (Research Virtual Machine) is used to study the reference-count of Java objects. It is designed to execute Java programs that are typically used in research on fundamental virtual machine design issues [9].

Our experiment is mainly targeted on the reference-counting collector. We are interested in the reasonable maximum reference-count of java objects. Through the experiment of running JVM SPEC 98 [17] in RVM, the reference-count of each object is obtained. We have modified the object header of the RVM in order to record the maximum reference-count of the object. The maximum value of the maximum reference-count is set to 255. If it is overflow, the reference-count will be reset to zero, and the object is subject to garbage collection.

According to the measurement, the average value of the maximum references is small (<= 2). There are over 99% of objects of which the maximum reference-count is less than or equal to two. Therefore, four bits for reference-count for each object are sufficient to cover 99% of Java objects.

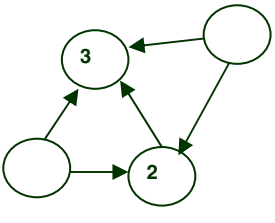


Fig. 3. Definition of reference count for Java objects

4 jHISC Architecture

In Java bytecode specification, there are several instructions defined to perform the tasks of the object-oriented features such as invokevirtual and putstatic. The ratio of

the object-oriented bytecodes (including access constant pool instruction and method call instructions) on SPEC JVM 98 benchmark is 31.87% [17]. Object-oriented instructions are added into jHISC instructions set in order to accelerate the object-manipulation operations by hardware support.

In jHISC architecture, three-address register-to register instruction form is used. jHISC processor have a register engine which contains 32 general registers of32 bits wide. Beside the 32-bit data, each register contains a 4-bit type field to describe the type of the data inside the register. During the process of the method invocations caused by ivk instruction, a method frame is allocated in the register. The method frame acts as the stack frame in Java virtual machine which is the method code working space. The register engine can be used to translate all the stack operations, which defined in Java bytecode, into 3 addressing jHISC instructions to accelerate the execution of Java applications.

5 Garbage Collection for Young Object

Generational garbage collection is used. The whole object heap is divided into two generation spaces. They are young space and mature space. All the new objects will be allocated in young space. Objects will be promoted into mature space after it is still alive in the young space for a period of time. In order to support real-time application, concurrent garbage collection will be used in both generation spaces. On this paper, we are focusing on the young space.

5.1 Reference Counting Object Cache

The main design concern of the memory management in young space is to provide fast object allocation. According to the research about the java object size and life, we have found that in most Java programs the vast majority of objects (often >95%) are very short-lived (i.e. are used as temporary data structures), small in size (90% objects =< 48 bytes). The object allocation occurs frequently.

The purpose of the scheme is to hold objects with their respective reference-counts in the cache to speedup the algorithm. The reference-count is modified directly in the cache to avoid accessing the main memory. For each instruction which will copy, delete or set a reference between two objects, it will check if the objects are held on the cache or not. If all the objects involved by this instruction are on the cache, the reference-count of these objects will be modified concurrently while the instruction is executing.

When the reference-count becomes zero, this implies that the object is not referenced by other objects and the running program cannot reach it. The cache line occupied by this object can then be reclaimed by the collector. When the reference-count of an object stays on cache overflow, it will be promoted to the mature space. Its reference-count field will be invalidated and it will be ignored by the reference-counting collector. Beside newly allocated instance objects, objects residing in the main

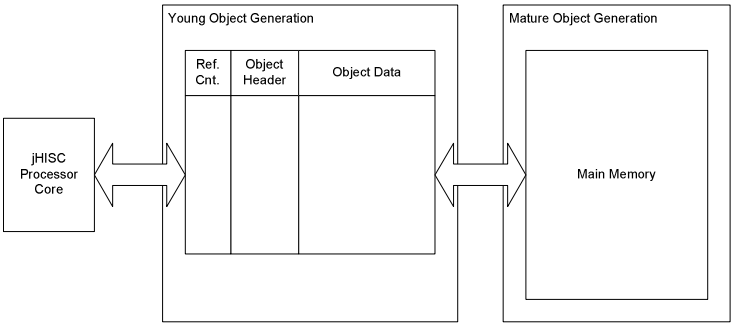


Fig. 4. Overview of jHISC Garbage Collection scheme

memory are allowed to be brought into the cache. In these cases, their reference-counts are set to all "1"s and they will not be collected by the reference-counting collector.

There is a 3-bit reference-count field to store the reference-count. Zero represents that the object is garbage and all '1's represents overflow. There is an M-bit to represent that the object has references form the mature space's object.

One of the main drawbacks of reference-counting is its computational expense. Therefore, we only maintain reference-count of an object in the cache. After an object is moved out from the cache, it will not be collected by the reference-counting collector and the reference-count on the cache will be invalidated. It reduces the computational expense for adjusting the reference-count in main memory.

5.2 Object Allocation for Small Size Objects

Object creation takes place in the object-cache directly instead on the main memory. As the lifetimes of small-size object in Java are short and there are spatial localities in the memory allocation and deallocation patterns in Java, hardware object allocator is proposed to do the small block sizes memory allocations and deallocations. Many of objects will die in the cache, without ever being written to the main memory.

A separated object allocation bit vector and a decision tree are used to allocate an object in hardware level [6] [7]. Buddy System is used to find the appropriate contiguous free bits in a bit vector. This allocation scheme supports the objects sizes of power of two. In our instance object cache, each object will occupy a whole cache line. The young object space is divided into slots of same size. Design of the binary tree can be further simplified. The object allocator determines if there is an available slot or not. If so, the allocator locates the address.

Objects of data size smaller or equal to a cache line, are allocated in the cache directly by the hardware object allocator. For objects of size greater than a cache line, the allocation request will be transferred to the operating system. These objects will be allocated in the mature object space. A free list is maintained by the memory manager of the operating system. Suitable free space will be searched to adapt the allocation request.

5.3 Hardware Write Barrier for Reference Count Update

During the execution time of the user Java program, we need to keep track of the action of the reference assignment and deletion operations and update the associated object's reference-count. Write barrier method is used to keep track of this operation.

In jHISC, we integrate write barrier into the instructions. There are four instructions responsible for reference update between objects. They are oo.pfld, oo.psfld, oo.pifld and array.store. While these reference update instruction is executing, we can update the reference-count of the object concurrently. When executing oo.pfld, an instance variable will be modified. If the data type of the variable is in reference type, write barrier will be activated. If the assigned references and the reference we are overwritten are in the object cache, the write barrier will update their reference-count. If the destination object is in the mature space and the object described by the assignment object is in the cache, the M-bit in the reference count field is set to 1.

Beside the references between objects, reference exists between stack /registers (register frame of jHISC) and objects. Reference-count update must be done on stack operations. Reference-count update process is invoked after the local registers movement operations ("data.move" instruction) if the type of the register are "reference".

6 Evaluation

A nearly real-time garbage collection with reference-counting has been done successfully. The reference-counting collector reuses the memory cell for further object allocation immediately when the objects become garbage. Reclamation of short-lived garbage objects is done in-cache without requiring any accesses to main memory. One of the characteristics of reference-counting algorithm is that it can garbage collect the object as soon as the object becomes garbage.

Objects with relative age zero and size equal to or smaller than a cache line can be garbage collected on the cache and will not reside in the main memory. This kind of objects would be staying on object cache all their lives. The small-size size object which can fit into a cache line can be allocated on the object cache directly. In the object allocation process in the object cache, all objects are treated the same size which is equal to the size of each cache line. According to the object behavior we have studied, over 90% of objects (sizes smaller than or equal to 48 bytes) are allocated directly on the object cache. The objects which have zero relative age are all reclaimed by the reference-count garbage collector. There are 70% of objects dead on object cache.

Table 4. Percentage of small size and short-lived object on Java MIDP applications

Applications	Percentage of objects which the size is smaller than <= 48 bytes and relative age is equal to zero
WebViewer	83%
WormGame	72%
PhotoAlbum	43%
AudioDemo	76%
Overall	**68%**

7 Further Investigation

A concurrent garbage collector for small-size and short-lived objects is designed. This collector can be used as the collector on young space of generational garbage collection. A secondary collector is need on mature object space to collect the garbage object that cannot be reclaimed by the reference-counting algorithm such as cyclic objects and objects with many references. A heap compaction algorithm is another issue for future study. Fixed-size object allocation may cause the fragmentation in the object heap. Heap compaction is needed to remove the fragment in the heap. The existing solution is for secondary level garbage collection, and heap compaction is stop-the-world algorithm. For future investigation, we will concentrate to design a nearly concurrent algorithm to meet the real-time requirement for embedded systems.

Acknowledgements

The work described in this paper was partially supported by a grant from City University of Hong Kong (Strategic Research Grant Project No. 7001548).

References

1. Fong, A., S., HISC: A High-level Instruction Set Computer. In 7th European Simulation Symposium, pages 406-410. The Society for Computer Simulation, Oct 1995.
2. Mok Pak Lun, Richard Li, Anthony Fong. Method manipulation in an object-oriented processor. ACM SIGARCH Computer Architecture News archive. Volume 31, Issue 4, September 2003.
3. G.Chen, M.Kandemir, N.Vijaykrishnan, M.J. Irwin. PennBench: A Benchmark Suite for Embedded Java Proceedings of the 5th Workshop on Workload Characterization. Austin, TX, Nov, 2002
4. Paul R. Wilson. Uniprocessor Garbage Collection Techniques. In *International Workshop on Memory Management*, St. Malo, France, September 1992.
5. R. Jones, R. Lins. Garbage Collection Algorithms for Automatic Dynamic Memory Management. John Wiley & Sons.1996
6. Witawas Srisa-an, Chai-Tien Dan Lo and J.M. Chang. Active Memory Garbage Collector for Real-Time Java Embedded Deveices. IEEE Transactions on Mobile Computing, volume 2, No. 2, pp. 89-101. 2003.
7. H.Cam, M.Abd-El-Barr, and S. M. Sait. A high-performance hardware-efficient memory allocation technique and design. International Conference on Computer Design, 1999. (ICCD '99), pp. 274-276. IEEE Computer Society Press, Oct 1999
8. S. Blackburn and K. Kathryn. Ulterior Reference Counting: Fast Garbage Collection without a Long Wait. OOPSLA' 03 ACM Conference on Object-Oriented Systems, Language and Applications, October 26-23, 2003.
9. B.Alpern, C.R. Attanasio, A. Cocchi, D. Lieber, S. Smith, T. Ngo, J. J. Barton, S. F. Hummel, J. C. Sheperd, and M. Mergen. Implementing Jalalpeo in Java. In Processings of the 1999 ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages & Applications, OOPSLA'99, Denver, Colorado, November 1-5, 1999, volume 34(10) of ACM SIGPLAN Notices, pages 314-324, ACM Press, Oct. 1999.

10. H.Azatchi and E. Petrank. Integration generations with advanced reference counting garbage collectors. In International Conference on Compiler Construction, Warsaw, Poland, Apr. 2003.
11. D. F. Bacon and V. T. Rajan. Concurrent cycle collection in reference counted systems. In J. L. Kuden Knudsen, editor, Proceedings of 15th European Conference on Object-Oriented Programming Languages, New Orleans, Louisiana, January 15-17, 2003, volume 38(1) of ACM SIGPLAN Notices. ACM Press, Jan. 2003.
12. L. Peter Deutsch and Daniel G. Bobrow. An efficient incremental automatic garbage collector. Communications of the ACM, 19(9):522-526, September 1976.
13. Geoergo E.Collins. A method for overlapping and erasure of lists. Communications of the ACM, 3(12):655-657, December 1960.
14. Adele Goldberg and D. Robson. Smalltalk-80: The Language and its Implementation. Addison-Weslev, 1983.
15. Larry Wall and Randal L. Scheartz. Programming Perl, O'Reilly and Associates, Inc., 1991.
16. Sun claims it's winning developer space, http://www.theinquirer.net, May, 2004
17. Young-Min Lee, Byung-Chul Tak, Hye-Seon Maeng, and Shin-Dug Kim. Real-Time Java Machine for Information Appliances. IEEE Transactions, volume 46, issue 4, p.949-957. 2000.
18. SPEC JVM98. http://www.specbench.org/osg/jvm98/

An Effective Instruction Cache Prefetch Policy by Exploiting Cache History Information*

Soong Hyun Shin, Cheol Hong Kim, and Chu Shik Jhon

Department of Electrical Engineering and Computer Science,
Seoul National University, Shilim-dong, Kwanak-gu, Seoul, Korea
{shordan, kimch, csjhon}@panda.snu.ac.kr

Abstract. The hit ratio of the first level cache is one of the most important factors in determining the performance of embedded computer systems. Prefetching from lower level memory structure is one of the techniques for improving the hit ratio of the first level cache. This paper proposes an effective prefetch scheme for the first level instruction cache by exploiting cache history information. The proposed scheme utilizes two factors to improve the prefetch efficiency: the disparity of block size between memory hierarchies and continuous same page hits. According to our simulations, the proposed prefetching scheme improves the performance by up to 6.3%.

Keywords: Computer architecture, embedded processor, instruction cache, cache prefetching.

1 Introduction

Minimizing the average data access time is crucial in designing embedded computer systems. As the gap between processor cycle speed and memory access time grows, access time to lower level memory structure has increased dramatically. Accordingly, first level cache plays one of the most important roles in determining the performance of computer systems.

Many researchers have examined how to improve the hit ratio of the first level cache. Prefetching from lower level memory structure is one of the techniques to improve the hit ratio of the first level cache. Prefetching can be implemented by software or hardware. Software-based prefetching needs compiler supports to insert prefetch instructions selectively by analyzing the program. However, it is very difficult to predict the block to be prefetched precisely at compile time, resulting in passing up many prefetching opportunities. Contrary to the software-based prefetching, hardware-based prefetching can use run-time information to predict the block to be prefetched. However, hardware-based prefetching schemes require additional hardware overhead. In this paper, we propose a hardware-based prefetching scheme to improve the hit ratio of the first level instruction cache (iL1) with little hardware overhead.

* This work was supported by Brain Korea 21.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 57–66, 2005.
© IFIP International Federation for Information Processing 2005

The hardware-based prefetching schemes are classified into sequential prefetching schemes and non-sequential prefetching schemes. Smith proposed sequential prefetching scheme for the first time[1]. OBL (One Block Lookahead) scheme[1] initiates prefetching only for the sequential block. There are three ways in prefetching by OBL scheme: always-prefetch, prefetch-on-miss and tagged-prefetch. The always-prefetch scheme prefetches the next block on each reference, so it increases the number of cache lookups significantly. The prefetch-on-miss scheme simply prefetches the next block only when one block is required due to a cache miss. The tagged-prefetch scheme is a modified version of the prefetch-on-miss scheme. It associates a tag bit with every memory block, and the tag bit is used to avoid prefetching same block again. Dahlgren et al. proposed an adaptive sequential prefetching scheme, varying the prefetch degree (the number of blocks to prefetch) dynamically[2]. In this scheme, the proportion of useful prefetches to total prefetches is periodically calculated. The prefetch degree is controlled according to the proportion. Jouppi suggested a FIFO stream buffer, which contains prefetched blocks before they are brought into the cache, to avoid cache pollution due to prefetching[3]. Each block in the buffer is brought into the cache if it is referenced, then a new block is pushed into the FIFO stream buffer. The cache pollution is avoided by using this buffer. However, if a cache miss occurs and the requested block is not located in the head entry of the buffer, the buffer is flushed and new sequences of blocks are fetched to the buffer, resulting in no use of prefetched blocks. The fetch directed instruction prefetching scheme[4] proposed by Reinman et al., one of the non-sequential prefetch schemes, utilizes the branch predictor for predicting program stream. In this scheme, the prefetch unit prefetches blocks according to the predicted program stream. Similar to the fetch directed instruction prefetching scheme, the execution history guided instruction prefetching scheme[5] proposed by Zhang et al. utilizes execution history. It is done by correlating execution history with cache miss history. Batcher et al. proposed the cluster miss prediction scheme for embedded cpu instruction caches[6]. The scheme is for real-time networking applications, so they profiles the characters of the dedicated application traces.

In most prefetch schemes, there are two types of references to the lower level memory structure; actual lookup and prefetch lookup. The actual lookup, caused by a cache miss in the first level cache, services normal memory reference. The prefetch lookup is caused by a predicted memory reference. If the prefetch prediction is incorrect, the prefetch lookup is turned out to be unsuccessful. These unsuccessful prefetch lookups increase the contention to the lower level memory structure without any contribution to the hit ratio of the first level cache.

For the iL1, utilizing sequential prefetch is more effective than non-sequential prefetch[7], but sequential prefetches increase the traffic to the lower level memory structure significantly. To reduce the traffic, we propose a prefetch scheme which doesn't request the prefetch lookup to the lower level memory structure separately, but requests the missed block and the next block simultaneously when a cache miss occurs. It is done by simply doubling the block fetch size. Moreover, the proposed scheme improves the prefetch accuracy by tracing the memory page access pattern of executed instruction sequence.

The rest of this paper is organized as follows. Section 2 describes the motivation of this paper. Section 3 presents the proposed prefetching scheme. Section 4 describes our evaluation methodology and shows detailed simulation results. Section 5 concludes this paper.

2 Motivation

It is commonly regarded that the program control flows sequentially. Generally, branch instructions occupy only 0 – 30% of total instructions[8][9]. Therefore, most instructions are expected to be accessed sequentially in the iL1. We try to exploit benefits from this characteristic of the program. Fig.1 shows the probability according to the relative distance between continuously referenced cache blocks in the iL1, obtained from our simulations using SimpleScalar[10]. We simulated 22 benchmarks from SPEC CPU2000[9]. The relative distance in the graph denotes the distance between continuously referenced cache blocks in the iL1. For example, in case that $n+1$th instruction is referenced after nth instruction in the iL1, the relative distance is one if the $n+1$th instruction is within the next block to the block containing nth instruction. In this graph, we excluded the probability of accessing same cache block (the relative distance is zero). The probability (the nth and the $n+1$th instruction are within the same block in the iL1) is meaningless with respect to prefetching, because the needed block is already located in the iL1. As shown in Fig.1, the probability that the relative distance is one is over 70%. From this result, we realized that the next block is referenced in the iL1 with very high probability after one cache block is referenced. Therefore, the hit ratio of the iL1 is expected to increase if the next block is prefetched from lower level memory structure when one block is requested due to iL1 cache miss. However, prefetching every next cache block is very expensive, because it may increase the bus contention significantly.

To find the way to maximize the prefetch efficiency, we also observed the relative distance between continuously referenced blocks in case that two blocks

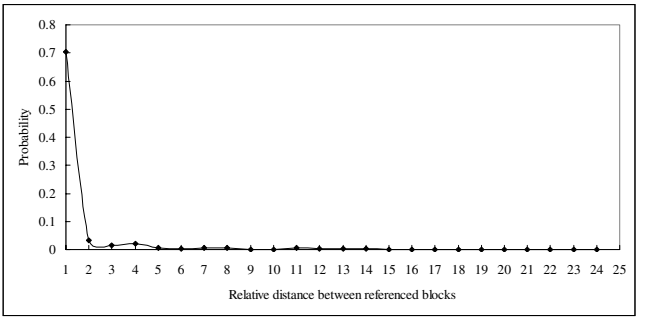


Fig. 1. The probability according to the relative distance between instructions executed. (Instruction cache block size=32B, Memory page size=4KB).

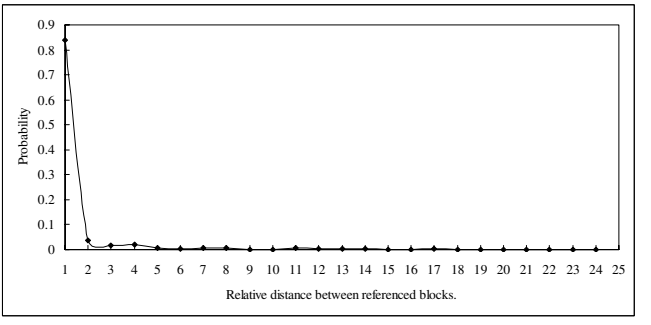


Fig. 2. The probability according to the relative distance between instructions within the same page. (Instruction cache block size=32 B, Memory page size=4KB).

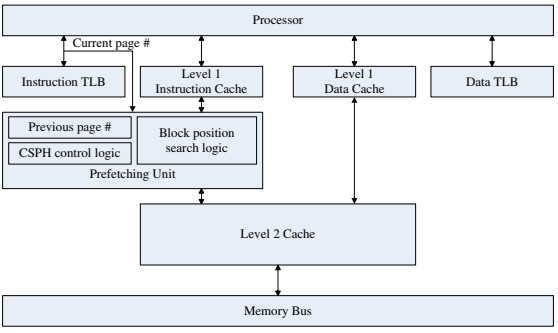
are within the same page. As shown in Fig.2, the probability of accessing the next block is 84% when executed instructions are within the same page. As shown in Fig.2, if prefetch operations are permitted only when the same page is accessed, the probability of accessing the next cache block is increased by 14% (from 70% to 84%). We propose a prefetch scheme to utilize these characteristics of applications.

3 Prefetch on Continuous Same Page Access

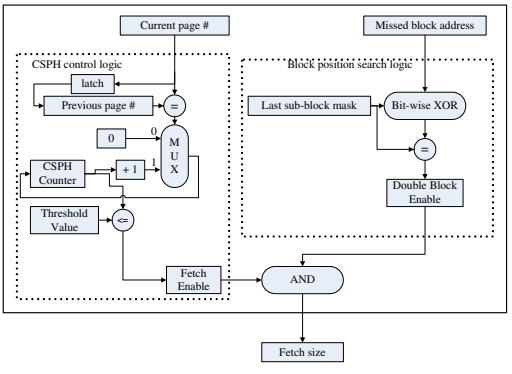
The *Prefetch on Continuous Same Page Access* (CSPA) scheme, proposed in this paper, initializes prefetch on misses if two conditions are satisfied: (1) the number of continuous same page hits (CSPH) reaches the threshold value, and (2) the block to prefetch isn't the last sub-block in the block of the lower level memory structure. Fig.3 depicts the proposed architecture. We assume two level cache architecture.

The simplest way to predict same page hit is to count the CSPH. If the CSPH exceeds the threshold value which we decide after simulations, we predict that the next instruction is within the page containing the previous instruction. The prefetching unit, placed between iL1 and L2, counts the CSPH to decide whether to prefetch or not. To count CSPH, the *previous page #* is recorded and it is compared with *current page #*. If these are equal, the $CSPH$ *counter* is increased. The $fetch$ *enable* signal is set when the $CSPH$ *counter* reaches the *threshold value*. (Fig.3(b)) This condition raises the probability of successful prefetch (Fig.1, Fig.2).

Most prefetch schemes have two types of references; actual lookup and prefetch lookup. For example, if the $A00$ (Fig.4(a)) block is required due to a cache miss in the iL1, the $A00$ is transferred to the iL1 and the $A01$, the next block of the missed block ($A00$), is prefetched by the prefetch lookup. For all prefetches, the L2 cache is looked up separately to the actual lookup. If prefetching the block is unsuccessful(the prefetched block is not referenced), the



(a) Overview of the architecture



(b) Precise depiction of the prefetching unit

Fig. 3. Proposed architecture and the prefetching unit

average memory access time increases by the cost of the prefetch. Even if the prefetch is successful, the memory traffic is not decreased because the prefetch lookup also looks up the L2. To remove this disadvantage, we initialize prefetch requests only when the missed block and the prefetched block are in the same block of lower level memory structure. It is because the L2 block has several iL1 blocks(generally, 4 sub-blocks) and it can service more quickly when requested iL1 blocks are sat in the same L2 block. Were it not so, the L2 must be accessed again to find the next sub-block. In CSPA scheme, two sub-blocks, $A00$ and $A01$, are requested simultaneously by the iL1 when the $A00$ is requested due to a miss in the iL1. If the prefetching block is successful, the prefetch lookup cost disappears and the memory traffic is decreased by removing prefetch lookup. In this scheme, if these two blocks aren't located in the same block (Fig.4(b)), fetching isn't initialized. In this case, only $A03$ block (without $A04$) is transferred to the iL1, because L2 has to be looked up again to find the block containing the $A04$.

As described, the CSPA scheme requests the missed block and the next block simultaneously when a cache miss occurs and two conditions(as described above) are satisfied. It is done by simply doubling the block fetch size. To make it possible, we assume that the $fetch\ size$ signal(Fig.3(b)) controls whether the L2 returns one sub-block or two sub-blocks.

It is easily noticed whether missed block is the last sub-block of the L2 cache block. The block offset of the L2 cache is divided into two fields: the lower field is a part overlapped by the block offset of the iL1, the upper field is the other part that indicates the sub-block in the L2 cache. Fig.5 shows an example. Memory address is divided into a tag, an index, and a block offset (Fig.5(a)). Fig.5(b) and Fig.5(c) depict the block offset masks of iL1 and L2, respectively. In this example, the iL1 cache block size is 32 bytes and the L2 cache block size is 256

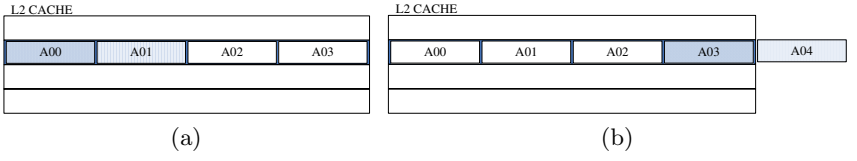


Fig. 4. Examples of allocation of a missed block and a prefetched block

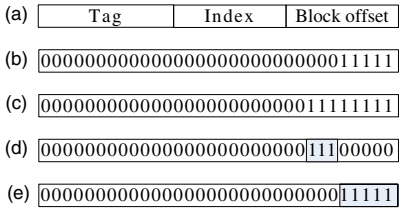


Fig. 5. Example of block offset masks and a last sub-block mask

bytes, that is, the bit widths of block offset are 5 and 8, respectively. The lower field of L2 block offset mask, shown in the Fig.5(e), is not used in the L2. L2 cache is searched by the upper field (Fig.5(d)) to find the sub-block which is requested by the first level cache. If the higher field is filled with '1', the requested block is the last sub-block of the L2 cache block. As shown in the Fig.5(d), if we apply XOR to the block offset of the iL1 and L2, the result is the $last\ sub-block\ mask$, indicated the upper field.

Consequently, through this operation, if the missed block is not the last sub-block, the *double block enable*(Fig.3(b)) is set. If all two enable signals, the $fetch\ enable$ and the *double block enable*, are set, $fetch\ size$ is set and the prefetch unit requests two blocks from the L2 cache(Fig.3(b)).

4 Performance Evaluation

4.1 Simulation Model

In order to determine the performance of the proposed prefetching scheme, we simulatated various benchmarks using SimpleScalar simulator[10]. Simulated applications are selected from SPEC2000 suite[9]. Simulated system parameters are shown in Table 1.

Table 1. System Parameters

System Parameter	Value
Processor	2-way superscalar processor
Level 1 instruction cache	16KB, 4way, 32bytes cache line, 1cycle latency
Level 1 data cache	16KB, 4way, 32bytes cache line, 1cycle latency, write-back
Level 2 cache	unified, 256KB, 8way, 256bytes cache line, 8cycle latency
Memory latency	32cycle latency + 1 cycles/8bytes

4.2 Simulation Results

At first, we simulated to determine the threshold value for the CSPH. Fig.6 shows the iL1 cache miss ratio and normalized execution time for various threshold values. The execution time is normalized to no-prefetch scheme. In the graph, 'CSPA-n' means that the threshold value is n. As shown in Fig.6, the CSPA-1, CSPA-2, and CSPA-4, show better performance than the others. We choose '4' for the threshold value among 1, 2, and 4 because the total number of prefetched instructions in the CSPA-4 scheme is less than the others. If the threshold value is too high, the performance degrades. As shown in Fig.6, as the threshold value increases, the iL1 miss ratio and the execution time increase. It is because, the successful prefetch ratio increases slightly as the threshold value increases, but

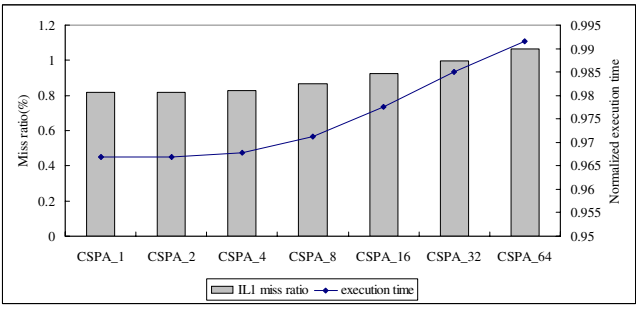


Fig. 6. Level 1 instruction cache miss ratio and normalized execution time. The execution time is normalized to no-prefetch scheme.

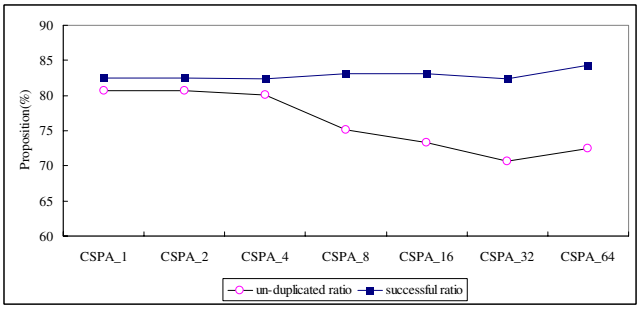


Fig. 7. The un-duplicated ratio and the successful ratio of prefetch for various prefetch configurations

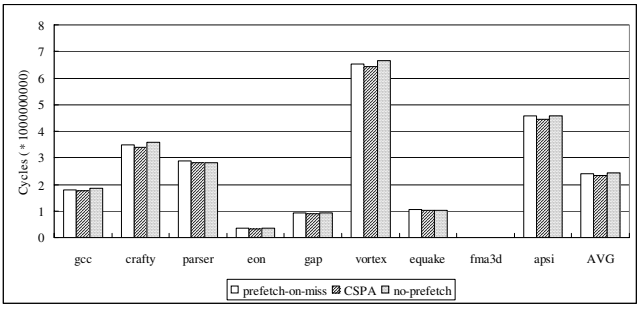


Fig. 8. The execution time for SPEC2000 benchmarks

the un-duplicated ratio decreases conspicuously as shown in Fig.7. The successful prefetch ratio is the proportion that prefetched block is used in the iL1 before it is evicted from the iL1, and the duplicated prefetch ratio is the proportion that prefetched block is useless because the block is already located in the iL1. It is because while the CSPH increases (instructions within the same page are referenced continuously), the probability increases that the prefetched blocks are already referenced. Besides we found the probability of accessing the next block is 84% when executed instructions are within the same page, as shown in the Fig.2, and the successful prefetch ratio of the CSPA is 82 – 84%. It means that the CSPA utilizes the sequential characteristic of benchmarks so effectively to maximize the prefetch efficiency and the successful prefetch ratio doesn't increase any more.

We compare the CSPA with the no-prefetch scheme and the prefetch-on-miss scheme, because the CSPA needs hardware overhead comparable to that of these two schemes, and other schemes like[4], [5], and [6] uses tables to keep informations which are heavier than the hardware cost of CSPA. In the iL1, the sequential prefetch scheme outperforms the non-sequential scheme[7] and hardware cost of sequential prefetch is less than that of non-sequential prefetch.

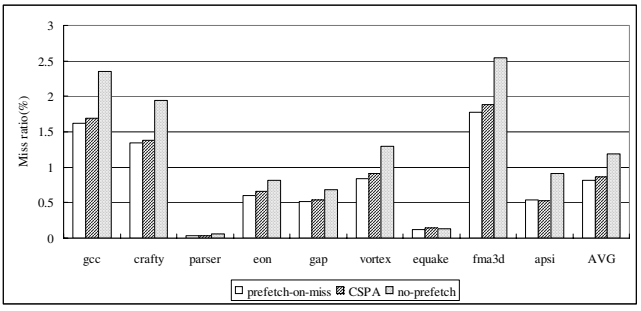


Fig. 9. The miss ratio for SPEC2000 benchmarks

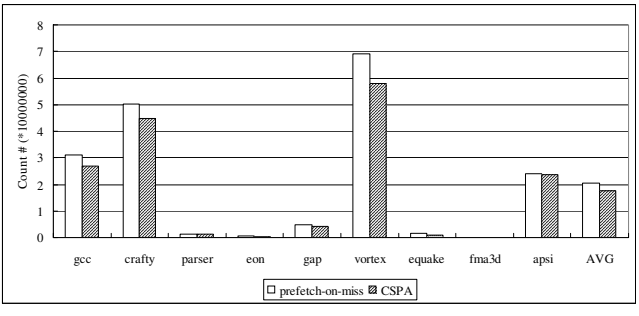


Fig. 10. The total number of prefetch for SPEC2000 benchmarks

Fig.8 and Fig.9 show the execution time and the miss ratio in the iL1. As shown in Fig.8, the performance of the CSPA is superior to other schemes. The CSPA outperforms the no-prefetch by 3.3% on the average and especially in the benchmarks, crafty and equake, the performances increase 5.3% and 6.3%,respectively. The CSPA also outperforms the prefetch-on-miss by 2.4% on the average. The miss ratio of the no-prefetch is 1.14% and that of the CSPA is 0.82%. The CSPA reduces miss ratio by 0.3%. Though the miss ratio of the prefetch-on-miss is less than that of the CSPA, the execution time of the CSPA is better. It is because the prefetch-on-miss performs prefetches 14% more than the CSPA, as shown in Fig.10. It is shown well in the benchmarks, gcc, crafty and vortex. As compared with the prefetch-on-miss scheme, the miss ratio of the CSPA is higher by 0.1% and the execution time is less by 2%. The reason of this conflicting result is that the CSPA reduces the total number of prefetch by 15.6%.

5 Conclusions

We have proposed the prefetch on continuous same page access scheme, which utilizes the disparity of block size between iL1 and the lower level memory structure and continuous same page hits. The simulation shows that the CSPA scheme

outperforms the no-prefetch policy by 3.3% on the average. The CSPA outperforms the prefetch-on-miss, and moreover it reduces the total number of prefetch by 14%. Simulation results show that the CSPA scheme is the most effective iL1 prefetching scheme among three compared schemes.

References

1. Smith, A.: Cache memories. ACM Computing Surveys **14** (1982) 473–530
2. Dahlgren, F., Dubois, M., Stenström, P.: Fixed and adaptive sequential prefetching in shared memory multiprocessors. In: International Conference on Parallel Processing. (1993) 56–63
3. Jouppi, N.P.: Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers. In: International Symposium on Computer Architecture. (1990) 364–373
4. Reinman, G., Calder, B., Austin, T.: Fetch directed instruction prefetching. In: International Symposium on Microarchitecture. (1999) 16–27
5. Zhang, Y., Haga, S., Barua, R.: Execution history guided instruction prefetching. In: International Conference on Supercomputing. (2002) 199–208
6. Batcher, K., Walker, R.: Cluster miss prediction with prefetch on miss for embedded cpu instruction caches. In: International Conference on Compilers, Architecture, and Synthesis for Embedded Systems. (2004) 24–34
7. Hsu, W.C., Smith, J.E.: A performance study of instruction cache prefetching methods. IEEE Transactions on Computers **47** (1998) 497–508
8. Lee, C., Potkonjak, M., Mangione-Smith, W.H.: Mediabench: A tool for evaluating and synthesizing multimedia and communicatons systems. In: International Symposium on Microarchitecture. (1997) 330–335
9. SPEC: (SPEC CPU2000 Benchmarks) http://www.specbench.org.
10. Burger, D., Austin, T.M., Bennett, S.: Evaluating future microprocessors: the simplescalar tool set. Technical Report TR-1308, Univ. of Wisconsin-Madison Computer Sciences Dept. (1997)

Efficient Switches for Network-on-Chip Based Embedded Systems

Hsin-Chou Chi and Chia-Ming Wu

Department of Computer Science and Information Engineering,
National Dong Hwa University,
Hualien, Taiwan
hcchi@mail.ndhu.edu.tw

Abstract. System-on-a-chip (SoC) has emerged to become a cost-effective approach for embedded systems design with rapid advance of semiconductor technology. It allows designers to integrate a number of heterogeneous IP blocks together based on a system interconnect. However, traditional dedicated wiring as the system interconnect has many shortcomings, such as non-scalable global wire delay, failure to achieve global synchronization, and errors due to signal integrity issues. These problems can be mitigated by the network-on-chip (NoC) architecture based on regular on-chip communication networks. In this paper, we present three efficient switch designs for NoC systems based on circuiting switching. Such switch designs with efficient buffer management can provide the on-chip network with guaranteed throughput and transmission latencies.

1 Introduction

System-on-a-chip (SoC) designs provide designers to integrate a number of IP blocks together. These IP cores can be a processor, DSP, FPGA block, or embedded memory. With rapid advance of semiconductor technology, the complexity of such SoC increases as a result. By the end of this decade, a single chip will accommodate up to 1-billion transistors. Thus, the number of IPs in SoC designs can scale from a few dozens to several hundreds or even thousands [1], [5].

One of the challenges in the billion-transistor era is the communication infrastructure between heterogeneous cores having different characteristics. The interconnect in current SoC based embedded systems for connecting IPs is typically dedicated wires or shared buses. The dedicated wiring approach provides the best communication performance, but its design has poor reusability and scalability. Furthermore, the wire latency and noise significantly affects the reliability of systems when the system complexity increases and the feature size decreases. The shared bus architecture provides a pool of bandwidth among all the cores in the system. However, the shared bus limits the growth of system complexity. To deal with the interconnect problems in SoC chips, a new methodology has to be developed for the next-generation SoC paradigm [2], [3], [4], [5].

To overcome the communication problems of SoC designs, network-on-chip (NoC) with regular tile-based architectures has been proposed for interconnecting the

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 67 – 76, 2005.
© IFIP International Federation for Information Processing 2005

IPs in SoC. Such on-chip interconnection networks provide a high-performance chip-level communication infrastructure with regularity and modularity. Fig. 1 shows the communication infrastructure of NoC with a 2D-mesh tile-based architecture [2], [3], [4], [5].

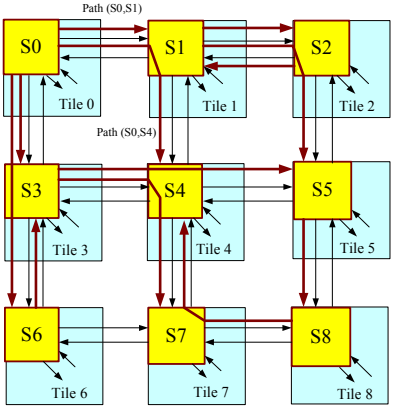


Fig. 1. The communication infrastructure of NoC with a 2D-mesh tile-based architecture. The dotted lines indicate some of the circuit paths provided by the circuit-switched network.

The major component in the communication infrastructure of NoC is the routing switch. In Fig. 1, each switch connects up to four neighboring switches and an associated local tile. The switch forwards packets from an input port to the destined output port according to the routing result of the network. In many applications such as multimedia, mobile component, telecommunication and consumer electronics, the characteristics of the communication traffic between IPs can be diversified. The switch needs to efficiently utilize the limited bandwidth of the links to deliver communication data between IPs and satisfies the different requirements of IPs. However, VLSI design is very cost-sensitive, and the required communication quality-of-service (QoS) must be achieved at a reasonable cost.

The circuit-switched network is suitable for NoC architectures since the transmission latency and throughput can be guaranteed. Furthermore, the complexity of the switch architecture in the circuit-switched network is low since its main function is to connect the incoming links to the outgoing links. The drawback of circuit-switched networks is locking resources for the duration of the data transfer. However, this can be alleviated by using virtual channel networks.

In this paper, we address a pre-scheduled circuit-switched network. Such network is suitable for the on-chip network. We also design and implement three efficient switches for such on-chip network. Compared with the design proposed in [9], our first design provides a lower cost. Furthermore, for decreasing the cost and latency, we design other two switch architectures. Our design provides high performance at a reasonable cost.

The next section describes the architecture of the circuit-switched network briefly. In Section 3, we present the design of our switches. Section 4 shows the implementation results. Finally, concluding remarks are given in Section 5.

2 Circuit-Switched Networks for SoC

When a sender wants to deliver data to a receiver in a circuit-switched network, the communication path between the sender and the receiver is established firstly. If there are lots of communication paths in the network at the same time, a link may be shared by multiple paths. Fig. 1 has shown such a network for example. In this diagram, the dotted lines represent the communication paths. The two paths *path(S0,S1)* and *path(S0,S4)* share the link between *Tile0* and *Tile1*. Time-division multiplexing (TDM) technique can solve the problems of sharing links. In TDM, a frame is divided into a fixed number of time slots. When a communication circuit is established across a TDM link, the network allocates a certain number of time slots in each frame of this circuit for the sole use of data transmission.

Before transmitting data, traditional computer networks use the routing probe to establish communication circuits. The routing probe progresses toward the destination to reserve a certain number of time slots for a communication circuit as it is transmitted through intermediate switches. When the probe reaches the destination, the communication circuit has been set up and an acknowledgement is transmitted back to the source. The circuit is released as the data has been transmitted. Fig. 2 shows the time-space diagram of a circuit-switched message. In such architecture, switches need to allocate the valid time slots for the circuit setup. If the network is congested, the setup time t_{setup} might be unacceptable. The traditional computer networks use this approach to achieve guaranteed service, since the communication paths in the cyberspace are too diverse to predict.

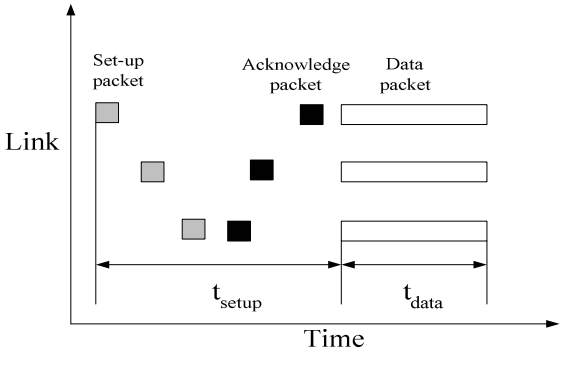


Fig. 2. The time-space diagram of a circuit-switched message

In NoC design, the requirements of communication between IPs are estimated before they are assigned to the tiles. The pre-established circuit-switched network is appropriate for NoC. In many methodologies of NoC approaches [4], [6], [7], [10], a mapping algorithm is needed to map IPs onto the tiles in the NoC architecture. The mapping algorithm must consider the communication delays and bandwidths between the tiles while it maps the IPs onto the tiles. Thus, after all IPs have been mapped, the communication paths among the tiles of the NoC are potentially existent. Switches

can use a table in the switch to store the routing information. Therefore, before transmitting data, switches do not have to set up a communication circuit.

There have been several recent studies on NoC architectures [2], [3], [4], [5], [8], [9], [11], [12], [13], [14], [15], [16], [17]. In [12], the proposed architecture differs from others in that they use the Butterfly Fat Tree architecture as the backbone of NoC. They also design a switch with wormhole routing for their NoC architecture. In [9], a guaranteed-throughput switch is presented. Their switch supports unicast and multicast with pre-scheduling. The input buffers and output buffers in this switch use several frame buffers implemented with random access memory. The cost of their design is too high to be useful in the on-chip network.

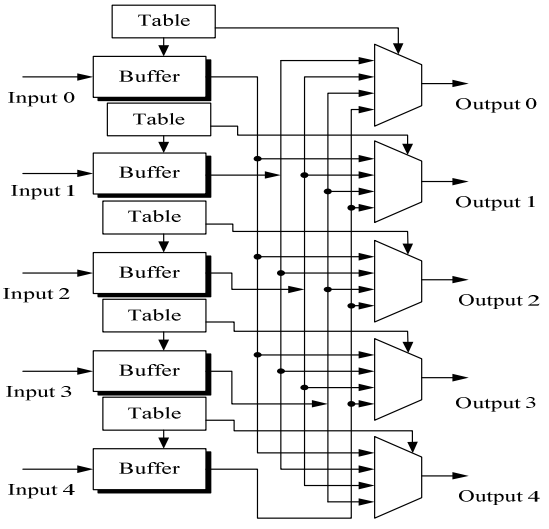


Fig. 3. The diagram of input buffering architecture

Switches connect to up to five links, as *S4* in Fig. 1. An arriving flit may be stored in the buffer firstly. The output port of the switch selects a flit from the buffer and injects it into the network according to the routing table. In the buffering strategy, we distinguish *output buffering* and *input buffering* by the location of buffers inside the switch [13]. An input buffering architecture is shown in Fig. 3. In this architecture, the buffers are at the inputs of the switch. This architecture needs a routing table to determine at which times which buffers are connected to which output ports with contention freedom. In this switch, flits from the input ports are buffered awaiting transmission by the output ports. The size of a frame buffer, FB, in the switch is:

$$FB = BW \times N \times S \tag{1}$$

where BW is the link bandwidth, N is the number of time slots, and S is the interval of the time slot. With a link bandwidth of 1Gbps, the number of time slots of 16, and the interval of the time slot of 2μs, the size of the FB is more than 33Kb. The sizes of the buffer in previously proposed architectures are over than two frames [9], [13].

Since NoC architecture design is cost-sensitive, the buffer size must be decreased as much as possible. We hence propose the switch architecture with only a frame buffer at every input.

We show the diagram of this architecture with one input and its associated output port in Fig. 4. We omit the global signals such as *clocks*, *reset* and some stimulating signals in this diagram for simplicity. In this architecture, each input has a frame buffer (FB) and two *input slot address tables*, ISATa and ISATb. The FB is used to store the input frame data, while the ISATs are used to store the addresses of the input data in the FBs. Each output port has a private *input time slot table* (ITST), and the ITST has two fields, *input port* (IP), *input time slot* (ITS). Furthermore, each input port has a private *output port table* (OPT). The OPT and the ITST determine at which times which buffers are connected to their output port respectively.

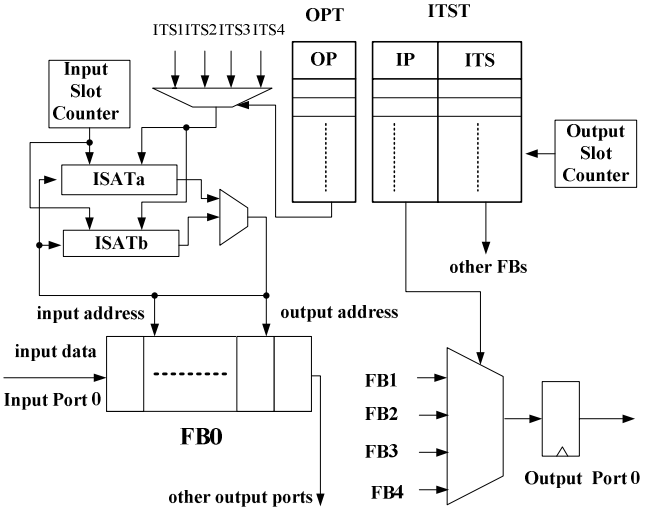


Fig. 4. The switch architecture with a frame buffer at each input

Latency is defined as the duration of a packet being transported from the sender to the receiver. The major factors of the latency are the latency of the switch and latency of the link, denoted by L_s and L_l. The latency for sending one bit of data from tile $tile_i$ to tile $tile_j$ can be analytically calculated as:

$$L_{bit}^{tile_i, tile_j} = \sum_{i \le n_{hops}} L_{S_i} + \left(n_{hops} - 1\right) \times L_l \tag{2}$$

where n_{hops} is the number of switches the bit passes on its way from tile $tile_i$ to tile $tile_j$. If the latencies of all switches in NoC are equal, the equation (5) becomes:

$$L_{bit}^{tile_i, tile_j} = n_{hops} \times L_s + \left(n_{hops} - 1\right) \times L_l \tag{3}$$

Compared with the architecture in [9], the architecture in Fig. 4 provides a lower cost. However, both switches do not deliver the data of the frame to the next destination until all flits of the frame have already arrived. Thus, the latency of these switches is too long. In NoC architecture, we do not need to do so. For decreasing the latency of the switch, the arriving data has to be transmitted as early as possible.

3 Design of Efficient Switches

In Fig. 5, the architecture is simpler than the architecture shown in Fig. 4. Arriving at an input port, the data is buffered in the temporary register. Without the whole frame has arrived, the data will be stored in the FB or directly sent to output ports. To do so, a comparator is needed to control the direction of the data stored of the temporary register. The comparator compares the input time slot of the data with the output address to generate the control signal which determines the destination of the data of the temporary register. If the data of the temporary register exactly is one of other output ports want to transmit, it is sent to this output port immediately. Otherwise, it will be stored in the FB, since it needs to wait for transmitting with a certain times. The address of the data in the FB is based on the input time slot of the data. The time of the data storing in the FB does not exceed the period of a frame transmission. Thus, a FB is enough to store all input data that need to wait for transmitting.

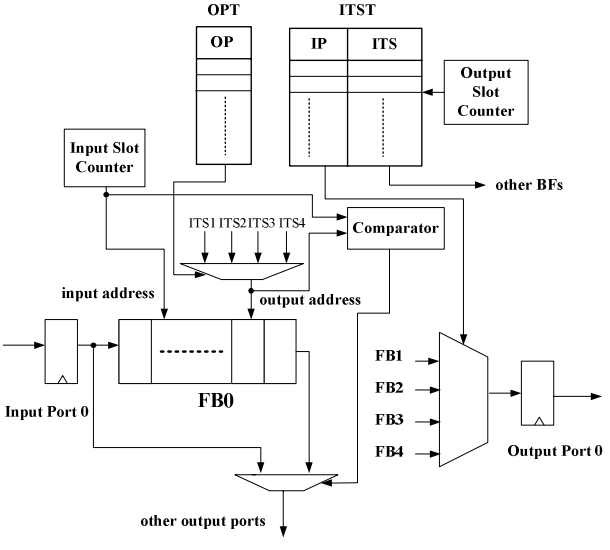


Fig. 5. The switch architecture with one FB at each input

A scheduler is needed to lower the waiting time. By employing optimal scheduling, the latency of data staying in switches can be minimized. Such optimal scheduling is an NP-complete problem. In the design described in the following, it meets the worst case scenario that all input data need to be stored. Therefore, the size of the buffer at each input is one FB.

In this architecture, some locations of the buffer will never be used for storing data, since some data is immediately transmitted. The unused space of the buffers in the switche is wasted. Though the size of the buffer depends on the scheduling algorithm, we also propose a second switch design in this section that reduces the cost of the switch. In this architecture, the buffer size at each input is less than a FB according to the application and can be designated by the designer.

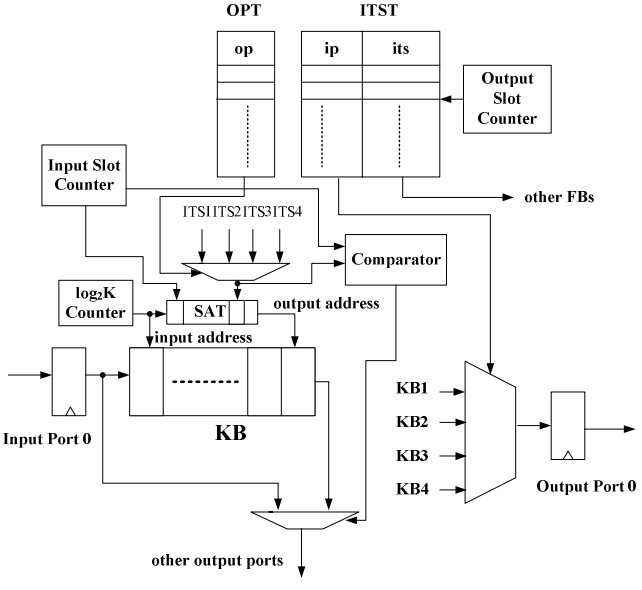


Fig. 6. The switch architecture with less than one FB at each input

Fig. 6 shows the low-latency and cost-effective switch architecture with less than one FB at each input. In this architecture, the size of the input buffer, denoted by KB, is designated by the design according to the specific application. The *slot address table* (SAT) stores the addresses of the data that needs to be buffered in the KB according to the input time slots of the data.

When arriving data needs to be stored into the KB, the $\log_2$K Counter provides an address. The size of the counter is at least $\log_2$K bits. The KB uses this address to store the data. Meanwhile, SAT stores this address in the location corresponding to the input time slot of the data. In each output time slot period, if one of the data in the KB is to be sent to the next switch, the SAT outputs the address of the data according to the ITS selected by the OP of the OPT. The data of this address in the KB is sent to the scheduled output port.

The size of the SAT is $S \times \lceil \log_2 K \rceil$ bits, where S is the number of time slots in a frame. With S of 32, K of 16, the size of the SAT is 128 bits. Such size is smaller than a location of the buffer in the switch. Hence, the cost of this architecture is significantly reduced.

4 Implementation

We have implemented the designs described in the above. We use pipelining technique to improve the performance. This architecture has three pipeline stages. The number of the time slots in a frame is 16. The data bus of the input port is 128 bits. We need one clock cycle for a packet crossing a link. Thus, the packet size is 128 bits. The table setup circuit is responsible for setting up the OPT and ITST and forwards the setup message to the next switch. Before initializing the system, the data for the setup table must be pre-scheduled and generated by a software scheduler. The scheduler could be part of the EDA tools for the NoC platform. On the other hand, we can also embed the scheduler into the chip and run be a processor core. Such design can dynamically schedule the circuits in NoC. Our design can support both scheduling methods.

Table 1 illustrates the chip area of three designs. Each is a 5 × 5 switch with a clock rate of at least 1 GHz supporting 16 time slots. The link width of the first and the third architectures is 128 bits while the second is 32 bits. The size of the buffer at each input of the switch for the first and the second architectures is a frame while the third is half a frame. In the experiment results, the third architecture reduces the cost significantly.

Table 1. Chip areas of three designs

Switch Architecture	Chip Area
128 bits with one frame buffer at each input	1.48mm^2
32 bits with one frame buffer at each input	0.4mm^2
128 bits with half a frame buffer at each input	0.26mm^2

Table 2. The comparison between our design and others

Design	Link Width	Freq.	A. Bandwidth	Area	Technology
Ours	128	1GHz	640Gb/s	0.26mm^2	0.18μm
[8]	32	185MHz	29.6Gb/s	3.5 mm^2	0.25μm
[13]	32	500MHz	80Gb/s	0.175mm^2	0.13μm

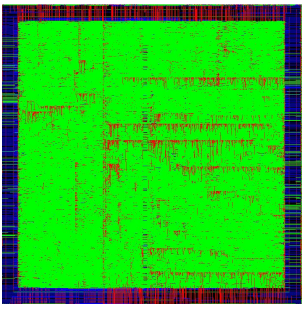


Fig. 7. The layout of the design of 128 bits with one frame buffer at each input

We have used TSMC 0.18μm technology to implement our designs. The layout of the first design in the table is shown in Fig. 7. Our chip has an aggregate bandwidth of 5 × 1 GHz × 128 bits = 640 Gbit/s. Table 2 shows the comparison between our design and others. In [8], a switch for multiprocessor with latency insensitive NoC is proposed. In [13], their switch also provides both circuit-switched and packet-switched transmissions. To reduce area, they have implemented a dedicated FIFO architecture with full-custom design. However, our design achieves higher throughput with less advanced technology.

5 Conclusions

In this paper, we propose the pre-scheduling circuit-switched network for the on-chip network in embedded system design. Such network does not set up communication paths before transmitting data. Thus, it provides guaranteed transmission throughputs and latencies. We have designed three efficient switch architectures to support this network. In these architectures, we show the trade-offs between hardware complexity and efficiency. We have used TSMC 0.18μm technology to implement our designs. The layout of one of the designs has an area of 0.26 mm^2 and offers 640 Gbit/s aggregate throughputs.

References

1. Semiconductor Industry Association, *International Technology Roadmap for Semiconductors*, World Semiconductor Council, 1999.
2. L. Benini, G. De Micheli, "Networks on Chips: A New SoC Paradigm," *IEEE Computer*, vol. 35, pp. 70 -78, Jan. 2002.
3. W. J. Dally and B. Towels, "Route packets, not wires: On-Chip Interconnection Networks," *Proceedings of 38th Design Automation Conference*, pp 684-689, June 2001.
4. S. Kumar, A. Jantsch, J.-P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, A. Hemani, "A Network on Chip Architecture and Design Methodology, " *Proceedings of the IEEE Computer Society Annual Symposium on VLSI, 2002 (ISVLSI.02)* , pp 105-112, April 2002.
5. A. Hemani, A. Jantsch, S. Kumar, A. Postula, J. Oberg, M. Millberg, and D. Lindqvist, "Network on Chip: An Architecture for Billion Transistor Era," *Proceedings of IEEE NorChip Conference*, pp. 166-173, Nov. 2000.
6. J. Hu and R. Marculescu, "Exploiting the Routing Flexibility for Energy/Performance Aware Mapping of Regular NoC Architectures," *Proceedings of Design Automation and Test in Europe Conference and Exhibition*, pp. 688-693, March 2003 .
7. T. Lei and S. Kumar, "A Two-step Genetic Algorithm for Mapping Task Graphs to a Network on Chip Architecture," *Proceedings of Euromicro Symposium on Digital System Design: Architectures, Methods and Tools, Turkey,* Sept. 2003.
8. L.-Y. Lin, C.-Y. Wang, P.-J. Huang, C.-C. Chou, J.-Y. Jou, "Communication-driven Task Binding for Multiprocessor with Latency Insensitive Network-on-Chip," *Proceedings of Asia and South Pacific. Design Automation Conference 2005.*
9. J. Liu, L.-R. Zheng and H. Tenhunen, "A Guaranteed-Throughput Switch for Network-on-Chip," *Proceedings of International Symposium on System-on-Chip,* Nov. 2003.

10. S. Murali and G. D. Micheli, "Bandwidth-Constrained Mapping of Cores onto NoC Architectures," *Proceedings of Design Automation and Test in Europe Conference and Exhibition*, pp. 869-901, Feb. 2004.
11. J. Nurmi, I. Saastamoinen, and D. Siguenza-Tortosa, "Interconnect IP Node for Future System-on-Chip Designs," *Proceedings of 1st IEEE International Workshop on Electronic Design, Test and Applications (DELTA '02)*, Jan. 2002.
12. P. P. Pande, C. Grecu, A. Ivanov, R. Saleh,"Design of a Switch for Network on Chip Applications," *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*, Jan. 2003.
13. E. Rijpkema, K. G. W. Goossens, A. Radulescu, J. Dielissen, J. Van Meerbergen, P. Wielage, and E. Waterlander, "Trade Offs in the Design of a Router with Both Guaranteed and Best-Effort Services for Networks on Chip," *Proceedings of Design Automation and Test in Europe Conference and Exhibition*, March 2003.
14. M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, and J. Rabaey, and A. Sangiovanni-Vencentelli, "Addressing the System-on-a-Chip Interconnect Woes through Communication-Based Design," *Proceedings of 38th Conference on Design automation*, June 2001.
15. S. Sathe, D. Wiklund, D. Liu, "Design of a Switching Node (Router) for On-chip Networks," *Proceedings of the 6th International Conference on ASIC (ASICON 2003)*, Oct. 2003.
16. D. Wiklund and D. Liu, "SoCBUS: Switched Network on Chip for Hard Real Time Embedded Systems," *Proceedings of International Parallel and Distributed Processing Symposium,* April 2003.
17. D. Wingard, "MicroNetwork-Based Integration for SOCs," *Proceedings of 38th Design Automation Conference*, June 2001.

An Efficient Dynamic Switching Mechanism (DSM) for Hybrid Processor Architecture

Akanda Md. Musfiquzzaman, Ben A. Abderazek,
Sotaro Kawata, and Masahiro Sowa

Graduate School of Information Systems,
The University of Electro-Communications,
1-5-1 Chofugaoka, Chofu-shi, 182-8585 Tokyo, Japan
akanda@sowa.is.uec.ac.jp

Abstract. Increasing the processor resources usability and boosting processor compatibility and capability to support multi-executions models in a single core are highly needed nowadays to benefit from the recent developments in electronics technology. This work introduces the concept of a dynamic switching mechanism (DSM), which supports multi-instruction set execution models in a single and simple processor core. This is achieved dynamically by execution $mode-switching$ scheme and a $sources-results$ locations computing unit for a novel queue execution model and a well-known stack based execution model. The queue execution model is based on queue computation that uses queue-registers, a circular queue data structure, for operands and results manipulations and assigns queue words according to a single assignment rule. We present the DSM mechanism and we describe its hardware complexity and preliminary evaluation results. We also describe the DSM target architecture.

Index Words: Hybrid processor, compatibility, design, dynamic switching mechanism, FaRM computing algorithm.

1 Introduction

Generally, the motivation for the design of a new architecture arose from the technological development, which changed gradually the architecture parameters traditionally used in the computer industry. With this growing changes, the computer architect is faced with answering the question what functionality has to be put on a single chip, giving them an extra performance edge. Nowadays, as we enter into an era of constant demand for faster and compatible processors as well as different Internet and network appliances using different processor architectures, it becomes extremely complicated and costly to develop a separate processor for every execution model that satisfy this demand. Internet applications, which are generally *stackbased* need high execution speed or high performance as defined by the literature. However, recently the term "high performance" is questioned again by many processor designers and computer users. Some consider that high performance means high execution speed or low execution time of some given

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 77–86, 2005.
© IFIP International Federation for Information Processing 2005

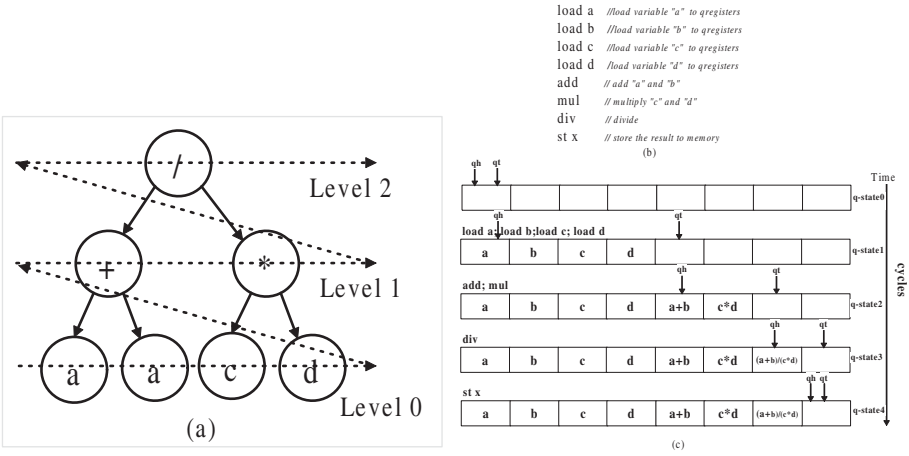


Fig. 1. Sample example of parallel execution in FARM Queue execution Model (FQM). (a) Data flow graph of the expression $X = (a+b)/(c*d)$. (b) QEM instruction sequence using Level order traversal, and (c) Queue contents at each execution state.

applications. Other define "high performance" differently. They consider that processors which support several execution models are the favourite candidates for high performance "award", since switching from processor to processor lead to difficulty and waste of time. This is true especially when users have different applications written for different execution models (Stack and RISC model for example). In this case, users are obliged to run these two applications separately on different machines. In conventional machines, this problem was *somehow* solved by a direct software translation techniques. However, these techniques still suffer from slow translation speed. Sun Microsystems proposed another alternative and designed its Stack-based Java processor, so that Java code can execute directly [4, 5]. According to its designers, the JavaChip-I, for example, is a highly efficient Java execution unit design. It delivers up to 20 times the Java performance for x86 and other general-purpose processor architectures, as well as up to five times the performance obtained by $just-in-time(JIT)$ compilers. It is evident that in term of reduced execution time (ET), the solution is better than the indirect way (translation) or the JIT scheme, but in term of compatibility, the processor still suffers from not being able to execute other codes. Therefore, reduced ET, compatibility, and cost are questioned again. Supported another execution model will eventually lead to more complex hardware. We realized, that supporting different instruction sets can yield superior operational attributes to those architectures that support a single instruction set. Our objectives are clear. First, we knew that to reduce die size and improve performance, DSM (Queue and Stack switching algorithm) would be implemented in the FaRM pipeline as a finite state machine rather than a traditional microcoded engine. Second, the solution would have to dynamically calculate Queue and Stack locations to architectural registers called shared storage unit (SSU). Thus avoiding the need for a translation stage. Finally, the resultant architecture would have to perform

16-bit fetches in order to fetch up to four instructions at once. To this end, we proposed a Hybrid processor architecture that addresses this and other problems as a pure-play architectural paradigm, which will integrate Stack and Queue execution models right into the FARM core[1, 2].

In this work we introduce the concept and the architecture of a dynamic switching mechanism (DSM), which supports multi-execution models in a single and simple processor core. This is achieved dynamically by execution *mode − switching* scheme and a *sources − results* locations computing unit for a novel queue execution model and a well-known stack based execution models.

This work has two major contributions:

- The first contribution is the proposal of a dynamic switching mechanism for multi-execution models support. To evaluate its efficiency, we designed it in register transfer level using Verilog HDL language. Then, we evaluated its functional correctness and its hardware complexity when compared with the Parallel Queue Qrocessor (PQP). Examples for FQM and FSM execution models are also given.
- Our second contribution, which is related to the first one, is a detailed architectural description of the target hybrid processor system and its complexity analysis

2 Hybrid Mode Execution Overview

The FARM architecture design is categorized by co-ordinates along a three-axis system. The three dimensions of the design space are: (1) number of instructions set supported by the architecture, (2) the storage scheme, and (3) the number of operands permitted for each mode[2]. The FARM system is a highly 32-bit processor that support a subset of the Queue instruction set (QEM) and Stack instruction set (SEM)[1].

The QEM mode uses a first-in-first-out Queue data structure as the underlying control mechanism for the manipulation of operands and results. In addition, the QEM is analogous to the stack execution model (SEM) in that it has operations in its instructions set, which implicitly reference an operand Queue, just as a stack machine has operations, which implicitly reference an operand Queue. Each instruction removes the required number of operands from the front of the Queue operand, performs some computations, and stores the result of computation into the Queue of operands at the specified offsets from the head of the Queue. The Queue of operand occupies continuous storage locations. A special register, called the Queue Head (QH), contains the address of the first operand in the operand Queue. Operands are retrieved from the front of the Queue by reading the location indicated by the QH pointer. Immediately after retrieving an operand, the QH is incremented so that it points at the next operand in the Queue. Results are returned to the rear of the operand Queue indicated by the Queue Tail (QT).

When switched for Stack-based mode, the switching circuitry and the functional computing unit (FCU) perform the job of execution model switching.

Hence, the FCU calculates the sources and destination for corresponding instructions.

In FSM, implicitly referenced operands are retrieved from the head of the operand stack and results are returned back onto the head of the stack. For example consider a *sub* instruction. In SEM model, the *sub* instruction pops two operands from the top of the stack (TOS), computes the difference and pushes the results back onto the top of the stack. In QEM mode, the *sub* instruction removes two operands from the front of the Queue (FOQ), computes their difference, and puts the results at the rear of the Queue (ROQ) indicated by the RQP. In the former case, the result of the operation is available at the TOS. In the later case, the result is behind any other operand in the Queue. This will have an enormous potential to effectively exploit pipelined ALU with a simple hardware, which, due to their hardware structure, normal SEM obviously cannot guaranty.

For better understanding of the novel QEM, we show in Fig. 1 a sample example that calculates a simple mathematical expression $x = (a + b)/(c * d)$. In the above example, the QEM instruction sequences were generated using the level order scan algorithm (LOST) published in our earlier work[2]. That is, in order to get instruction sequence that can be correctly executed in FQM, the data flow graph for the given expressing should be traversed from the deep level to the top level and from left to right. In Fig. 1(c) the queue contents (later reffered as shared storage unit) at each cycle are shown. As illustrated in Fig. 1(a) and Fig. 1(b),the first four instructions are independent. Hence , they are processed in parallel (q-sate1). The second level in the data flow graph also has two independent instructions (+ and *). These two instructions are also executed in parallel (q-state2). At the third level, only *div* instruction is generated and is separately executed (q-state3). The result is finally stored into the PROG/DATA memory (not shown in the figure).

3 DSM Mechanism Overview

The DSM mechanism is implemented in a so-named hybrid processor architecture (FARM)[1]. Before we describe the DSM mechanism and its functional description, we first give a brief architectural overview about the FARM core that adopts the DSM mechanism. The target processor is based on Consume-produce Order Queue Computational Model(FQC) and Stack computational model(FSC)[1]. The processor has six pipeline stages and is based on 16-bit instruction set architecture. It supports two execution modes: Queue mode (FQM) and Stack mode (FSM). The basic block diagram of FaRM is given in Fig. 2. The FARM core consists of the following units: (1) Instruction Fetch unit (FU), (2) Decode Unit (DU), (3) FaRM Computation unit (FCU), (4) Issue Unit (IU), (5) Execution Unit (EXE) and (6) Shared Storage Unit (SSU). The basic block diagram of FaRMqs is given in Fig. 2.

Instruction Fetch: The fetch unit fetches 4*16-bits instructions/cycle from the program memory and inserts them into the fetch buffer.

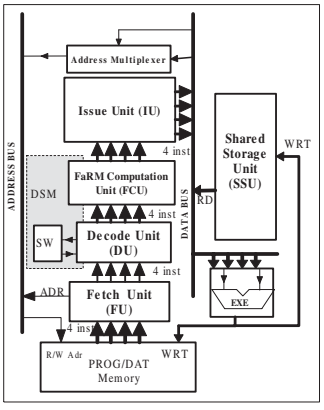


Fig. 2. Hybrid FARM system architecture

Instruction Decode: Decodes the instruction opcode and operands. The DU has four decode circuits (DC) and one Mode Selector Register (MS) that is set to zero or one according the type of instruction being decoded.
FaRM Computing Unit: The FaRM Computation unit gathers information mainly from the decode unit and uses them to compute the instruction sources and destination locations for FQM and FSM models.
Issue stage: The Issue stage issues ready instructions to the execution unit. Memory and registers dependency are checked by this unit/stage. This unit also checks the sources availability for each instruction.
Execution Unit: The EXE executes issued instructions and sends the results to the SSU or the data memory. The EXE consists of: (1) Arithmetic logical unit (ALU), (2) Shift unit, (3) Set register unit, (4) Load/Store unit, (5) Move and Compare unit and (7) Branch unit.
Write back Unit: The write back unit writes the result back to the PROG/DATA memory or the shared storage unit (SSU). The SSU is a 32*256 registers and is shared by both FQM and FSM execution models. In FQM, the SSU behaves as a conventional register file-like. However, in FSM, the system organizes the SSU access as a last-in-first-out.

4 DSM Architectural Details

As we earlier mentioned, the DSM system consists of a switching circuitry (SW) and a dynamic computation unit (FCU). It is implemented in a hybrid processor architecture. The FCU unit calculates $on-the-fly$ the sources and destinations for instructions in both FQM and FSM executions modes. A block diagram of the DSM mechanism is illustrated in Fig. 3. The DSM detects the instruction mode by decoding the operand of the *switch* instruction. After the mode detection, it inserts a $mode-bit$ for all instructions between the current and the next *switch* instruction. Hence, the same operation can be used for both modes. This will

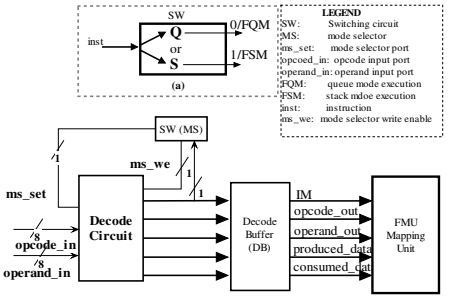


Fig. 3. DSM Basic Block Diagram

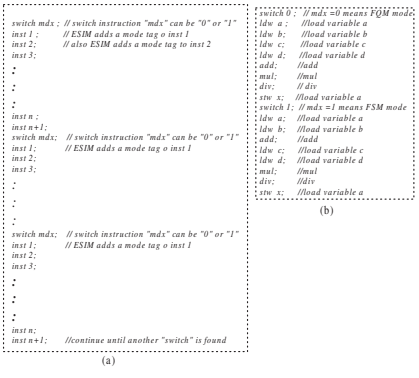


Fig. 4. FARM programming Models:(a) Programming Model, (b) Example of FQM and FSM assembly program

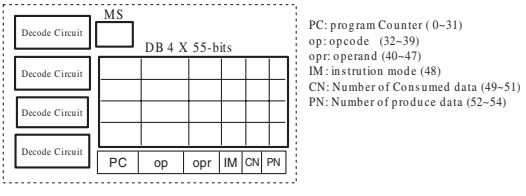


Fig. 5. DSM hardware components

increase the resources usability and the overall system performance (discussed later).

Dynamic switching example: In Fig. 4, we show the FARM programming model and an assembly program example. The first instruction at a given program should be always a "switch" instruction. This instruction will guide the decode unit to dynamically *reprogram* itself according to the *switch* instruction's operand value (*zero* for FQM mode and *one* for FSM mode). Instructions that follow *switch* instruction continue, then, execution in the FSM or FQM

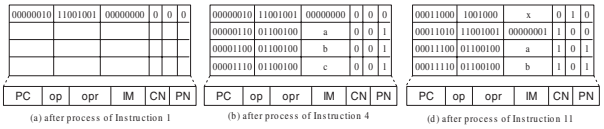


Fig. 6. Various DSM states over hybrid execution modes

until another *switch* instruction is found (refer to Fig. 4). The DSM (major hardware components) mechanism are illustrated in Fig. 5. This figure shows the initial state of the decode buffer (DB)and mode switch (MS) register before switch instruction is decoded. Fig. 6 shows the contents of the DB at different states for different instructions.

4.1 Dynamic Computation Algorithm

Dynamic Computation Algorithm (DCA) has two mapping algorithms: (1) FARM Queue Computing and (2) FARM Stack Computing.

FQM sources and destination computing: In order to have a correct execution, each instruction needs to know the values of the QH (Queue Head) and the QT (Queue Tail). The above values are easy to know in serial Queue execution model, since the QH is always used to fetch instruction from the operand queue (OPQ) and the QT is always used to store the result of the computation in to the tail of the OPQ. However, in the parallel execution scheme the above pointers are not explicitly determined. This is due to the fact that previous instructions are simultaneously executed and may not be completed in order. Fig. 7(a) shows the block diagram of queue computing circuits (FQC). The FQC is based on the Produce-consume order Queue Computation Mechanism.

The QH calculation for an instruction I(n+1) is given by:

$$QH(n+1) = QH(n) + (consumed_data) \tag{1}$$

Where, Consumed Data is the number of fetched operands of an instruction.

The QT calculation of an instruction I(n+1) is calculated by:

$$QT(n+1) = QTP(n) + (produced_data) \tag{2}$$

Where, Produced Data is the number of the results generated by an instruction

FSM sources and destination computing: FARM Stack Mechanism (FSM) is based on the pure stack Mechanism. For mapping the source and destination FSM used only one register named *Top*. Fig. 7(b) shows the block diagram of FaRM Stack (FSM) mechanism. The *Top* pointer calculation for an instruction (n±1)is given bellow:

$$TOP(n \pm 1) = TOP(n) - consumed_data + produced_data \tag{3}$$

Here, we use ± because top pointer does not only increase but also decrease. For *load* instruction *Top* pointer will increase and for *store* instruction *Top* will

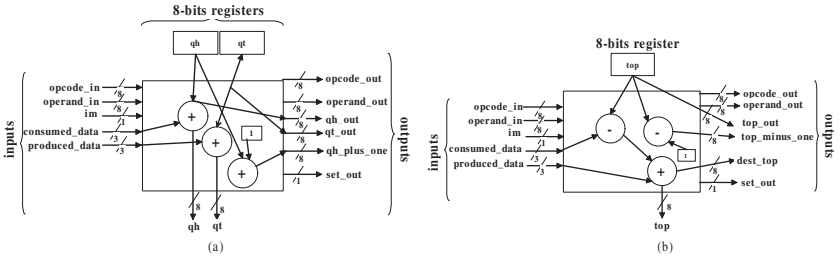


Fig. 7. FQM and FSM computing circuits: (a) FQM computing circuit; (b) FSM computing circuit

decrease. After assigning instruction, the FSM sends instructions to the FARM Computation Buffer (FCB).

Example for FQM Computing: To calculate the sources and destination location for FQM execution, we use equation (1) and (2) to assigning the QH and QT pointers value. To illustrate the idea, we use the simple expression $X = (a + b)/(c * d)$ as an example. The assembly code for the above expression in shown in Fig. 4(b). In FARM programming model, the first instruction should be always a *switch* type instruction. In this example, the operand of the *switsh* is 0 which indicates that the being loaded program should be executed in FQM. As a result, the mode-bit for all following instruction will be set to FQM mode.

Example for FSM Computing: We use formula 3in Stack Computing to calculate the TOP pointer value for each instruction. The assembly code of a simple expression $X = (a + b)/(c * d)$ is also shown Fig. 4. In this case, the operand of *switch* instruction is *one*, which indicates a stack program instructions sequence. The DCA uses the stack mapping mechanism. For this case, the DCA sets a "*setsignal*" for each instruction ("1" indicates the stack instructions) and requests the issue stage for sequential issue.

5 DSM Implementation Evaluation

We designed the DSM mechanism in register transfer level using Verilog HDL. In order to evaluate the effectiveness and the correctness of the mechanism, we performed two types of evaluations: (1) Functional evaluation level, (2) Hardware evaluation level.

Functional verification: We described the DSM in RTL level and used Verilog-XL and Simvision simulators to evaluate the functional simulation result. We captured the input and output signals changes for several cases.

Hardware level evaluation : The DSM and the whole system core were designed in RTL level and correctly integrated and synthesized in a Altra STRATIX-EP1S25F1020 device. The design results for area and speed optimizations are shown in Table 1. Notice that the DSM is integrated in the DU

Table 1. Hybrid FARM processor complexity (in flip-flops) evaluation results over speed and area optimizations. The DSM mechanism is implemented on Decode and FCA units. SOPT means speed optimizations; AOPT means area optimizations.

Description	Unit Name	SOPT	AOPT
Fetch Module	FU	201	166
Decode Module	DU	1039	975
Computation Module	FCA	848	800
Issue module	IU	589	326
Execution module	EXE	1140	1140
Shared storage module	SSU	6545	5730

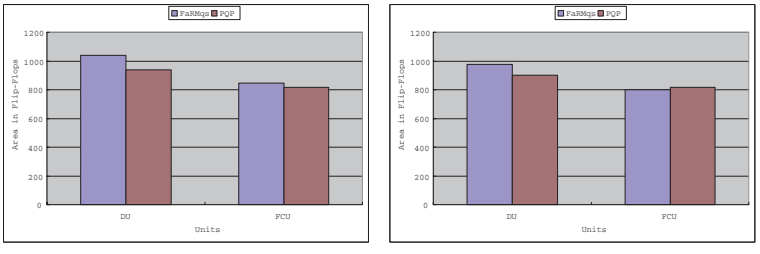


Fig. 8. Hardware Complexity evaluation and comparison: (a) The left side figure shows the area the comparison over Speed optimizations, (b) The right side figure shows the area comparison over Area optimizations

and FCU units. In order to know the real hardware complexity of the DSM, we compared the units that integrated it with other units in a similar queue processor architecture that do not include switching capability. From the above comparison, we found that the DSM mechanism uses about 60% of DU and 40% of the FCU. Fig. 8 shows the comparison results of the area with a conventional Queue processor core over speed and area optimizations respectively. From the above simulation, we noticed that the additional area for DSM mechanism is acceptable for both speed and area optimizations. We found that the DU and the FCU use about 9.72% and 3.77% extra area when compared with conventional queue processor's decode and queue computation modules.

6 Concluding Remarks

In this paper, we have proposed a dynamic switching mechanism (DSM) for a hybrid processor architecture that supports Queue and Stack computation modes in a shared resources single processor core. This is achieved dynamically by execution $mode-switching$ scheme and a $sources-results$ locations computing unit. We have presented the novel aspects of the DSM mechanism as well as the the architecture description of a hybrid processor system that adopts the DSM mechanism.

We designed the hardware of the DSM and verify its functionality and overall complexity. From the design and evaluation results, we proved that the DSM mechanism can be used without enormous additional hardware when compared with conventional queue processor core (about 9.72% for DU and 3.77% for FMU extra hardware when compared with Queue processor core). We also conclude that the overall performance of the hybrid FARM architecture, which is still in its infancy, is acceptable. Hence, the DSM system increases the resources usability and the overall performance of the FARM processor.

Our future work is to evaluate the whole system with real benchmark programs and estimate the real complexity of the whole system.

References

1. M. M. Akanda , B. A. Abderazek , T. Yoshinaga, M. Sowa, "FaRMqs: Hybrid Processor Architecture in Verilog HDL" IPSJ, 67th Conf., March 2 3, 2005. for Parallel Execution in Parallel Queue Processor", IPSJ, 66th Conf., March 9 11, 2004, pp.1-69 1-70.
2. B. A. Abderazek, Kirilka Nikolova, and M. Sowa: "FARM-Queue Mode: On a Practical Queue Execution Model", Proceedings of the Int. Conf. on Circuits and Systems, Computers and Communications, Tokushima, Japan, pp.939-944, July 2001
3. M. Sowa, B. A. Abderazek, S. Shigeta, K. Nikolova, and T. Yoshinaga, " Proposal and Design of a Parallel Queue Processor Architecture (PQP) ", 14th IASTED Int. Conf. on Parallel and Distributed Computing and System, Cambridge, USA, pp. 554-560, Nov. 2002.
4. N. VijayKrishnan, "Issues in the Design of JAVA Processor Architecture". PhD dissertation, University of South Florida, Tampa, FL-33620. December 1998.
5. R. Radhakrishnan, N. Vijaykrishnan, L. John and A. Sivasubramanium, "Architectural issues in java runtime systems," Tech. Rep. TR-990719, 1999.
6. Sowa Laboratory, www.sowa.is.uec.ac.jp

Design of Face Recognition Door Manager System Based on DSP

Dongbing Pu[1,2], Changrui Du[1], Zhezhou Yu[1], and Chunguang Zhou[1,*]

[1] Department of Computer Science, Jilin University, Changchun, 130012, China
[2] College of Computer, Northeast Normal University, Changchun, 130024, China
Jilin University, Changchun, 130012, China
cgzhou@jlu.edu.cn

Abstract. This paper focuses on the design of hardware framework with DSP based on embedded face recognition door manager system including the detailed technology of circuit design among the image sensor, DSP, the door manager controller, FLASH, SDRAM, RJ45 port, timer and reset. At the same time, both the programs and algorithms are discussed in the paper. The program on face recognition has been tested on the designed board; it runs all right with fixed light condition. Not only the size and cost of the board are small, but also it is fast and highly reliable.

1 Introduction

The Digital Signal Processor (DSP) is a kind of specialized microprocessor designed for realization of digital signal processing quickly in the real time, which adopts the Harvard structure. [1] It has some good characteristics such as the convenient interface, the good stability, the high accuracy, the good reusability and convenient integration etc. [2] Along with a fast development of DSP, its function is more and more stronger, the function price ratio rises continuously, the method of development has been improved continuously, so it also quickly becomes the core component of numerous electronic products. It is extensively applied to many fields such as the correspondence electronics, signal processing, automatic control, radar, military, aviation, medical treatment, home appliances, electric power and electronics etc. On the other hand, the traditional methods of identification such as passwords, certificates, IC card etc, which often cause the falsification, theft and crack as they can be separated from the owner. They couldn't fully be satisfied with the modern social economic activities and social safety. Therefore, the personal identification system based on the human body biologic characteristic gradually becomes very well known to the society. But the traditional face recognition system are achieved depending on large-scale integrated circuits same as microcomputers, both drivers of peripherals and image capturing depended on the hardware and operating system in a computer, which result in some disadvantages such

* Corresponding author.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 87 – 97, 2005.
© IFIP International Federation for Information Processing 2005

as the slow image processing, low reliability, poor function in real time etc. [3] Applying DSP and the image sensor to the body identification system with door manager system can achieve such advantages as quick speed, low cost and considerable reliability. This paper focuses on the design of hardware framework in a small face recognition door manager system, which based on DSP and image sensor. Such problems as the peripherals, interface design and the program of face recognition also have been discussed elaborately in the paper.

2 System Architecture Design

The architecture of the system is shown in Figure 1. BlackfinADSP-532 is the core of the hardware. On the other hand, the peripheral chips primarily consist of asynchronous FLASH, SDRAM, the image sensor, the door manager controller, network chip, clock, power, resets, and JTAG.

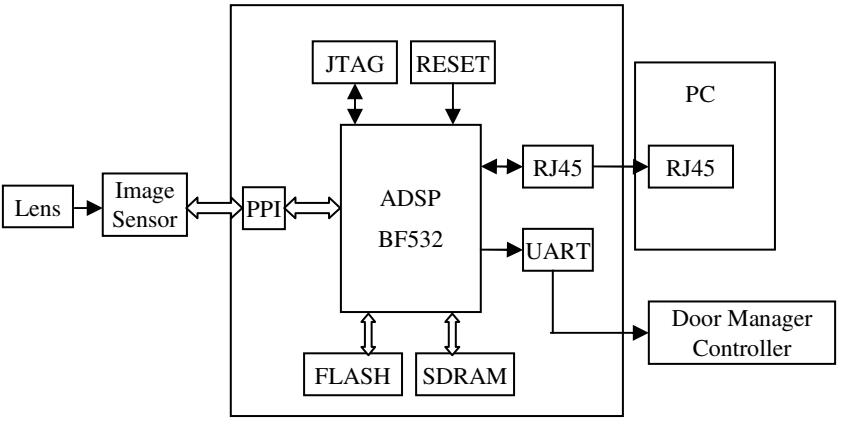


Fig. 1. Architecture of system

Firstly the system is initialized and boots along with electrifying, then it loads programs and data from FLASH to SDRAM. The data mainly includes the face detection and location template, mean face matrix, eigenface template and match template etc. When someone wants to pass the door, he or she gives an interrupt signal to the system by pressing the interrupt key. The image sensor captures the image via PPI interface under DSP controlling. Then the captured image is transmitted to the off-chip SDRAM. A feature value of person face will be achieved here when programs complete to process the image information. Since programs bring about a great deal of temporary data when obtain the feature value of person face, whereas the on-chip RAM for DSP is not enough to hold them. Thus an extended off-chip SDRAM of 4M*16-bit is added to the system. Those tasks including face detection and location, feature extraction is mainly processed in the extended SDRAM. The following task is matching the extracted feature value with data stored in the database of face feature,

and calculating the threshold value of similar degree. Eventually DSP sends message to the door manager controller through the UART interface according to the similar degree threshold value. The door manager controller can send message to open the door or play warning sound. Meanwhile, the log of recognition will be stored in server-named PC through RJ45 interface. These log information can be available for management. The JTAG of the system is used for emulating, debugging on-line and updating the information of person's face feature periodically.

3 Interface Design

3.1 Image Sensor Interface Design

In order to increase the speed of image capturing, the system adapts HDCS1020 image sensor. It can capture the image of CIF (352*288) up to 30f/s with A/D conversion itself. [4] And it can joint the data bus of DSP directly without any conflict.

Because programs of system process black/white image, so the data bus D0-D7 of image sensor connects with the PPI pins of DSP. DSP's GPIO pin can imitate the I2C protocol to control the image sensor.[5] Figure 2 shows the way of connection between DSP and the image sensor.

The control signals sent by DSP include two parts: address codes and control signals. Address codes are used to select the image sensor, and control signals perform the initialization and image capture. Control signals include beginning signal, response signal and ending signal. Beginning signal shows that SCL is high Data begin to be transmitted when SDA jumping from high to low. The image sensor sends response signal to DSP when it has already received 8-bit data. DSP sends signals to the image sensor and waits until response signals from the image sensor arriving to decide whether continue signal from it. When SCL is high and SDA jumps from low to high, DSP sends the end signal. Figure 3 shows the scheduling of I2C bus protocol. [6]

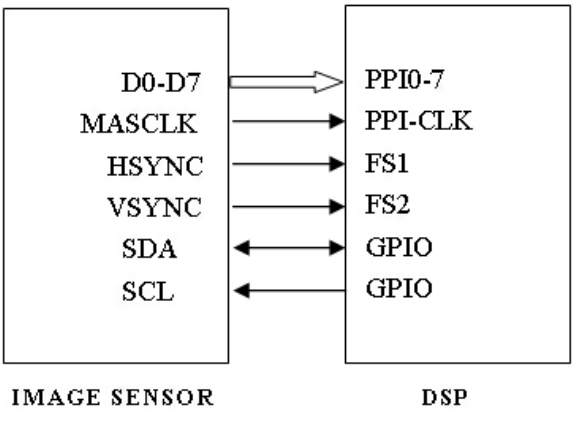


Fig. 2. Connection between image sensor and DSP

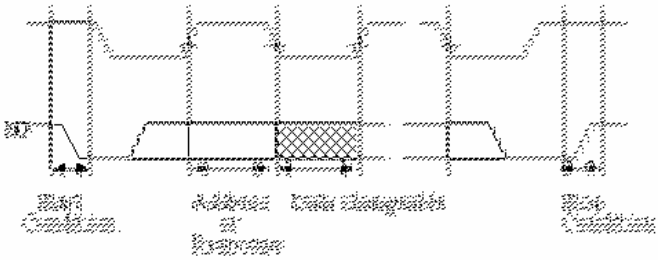


Fig. 3. Scheduling of I^2C bus protocol

3.2 FLASH Design

DSP uses a 16-bit EMIF loading model. In this model, pins of BOOTM [1:0] are set to 01. The boot kernel in the on-chip Boot Rom loads the programs and data from FLASH to SDRAM, and then the system begins to run. [7-8] The address corresponds to the BANKO space of DSP, so we connect $\overline{CE}$ of FLASH to $\overline{AMSO}$ of DSP. As the programs and data are enormous, we added an extended 4BANK*512K*16-bit of FLASH to the system. $\overline{AMSO-3}$ pins of DSP respectively connect to $\overline{CE}$ pin of four FLASH chips. A[19..1] are address lines of DSP, they are connected corresponding to A[18..0] of FLASH which is shown in figure 4. 4MB spaces of FLASH can be enough to hold system program, data and face feature templates of about 1000 persons.

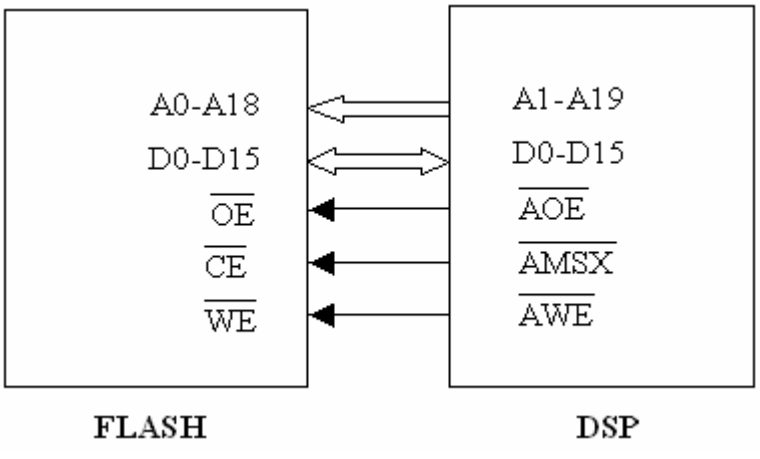


Fig. 4. Connection between FLASH and DSP

3.3 SDRAM Design

In order to provide a bigger space for programs and data of the algorithm, an extended 4M*16-bit of SDRAM is added to the system. As the row and line address are reusable for Blackfin, its connection way is greatly different from DSP 54X series of T1. The

connection way for boundary/row and column address is shown in figure 5. The row selecting $\overline{SRAS}$, column selecting $\overline{SCAS}$ and writable signal $\overline{SWE}$ of DSP are directly connected to the corresponding pin of SDRAM. A1-10, A12 and A18-19 address bus of DSP are respectively connected to A1-10, A11 and BA0-BA1 of SDRAM; SA10 and SDQM0-1 of DSP are connected to A10, DQML and DQMH of SDRAM respectively.[9]

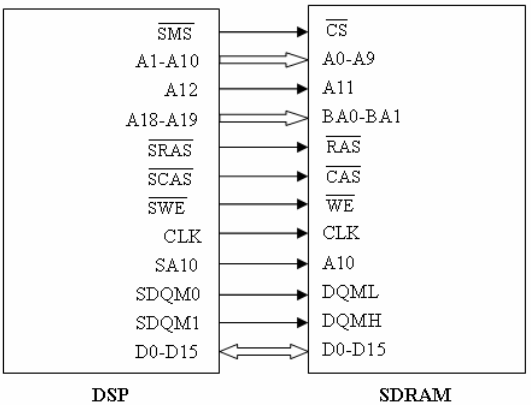


Fig. 5. Connection between DSP and SDRAM

3.4 Corresponding Interface Design Between DSP and PC

The LAN91C111 network chip is built in the hardware of our system. It is 3.3v MII (media independent interface) integrated the MAC+PHY on-chip and 128 pins QFP package. [10] The LAN91C111 integrates the IEEE802.3 Physical Layer (PHY) and Media Access Control (MAC) into the same silicon. The data path connection between the MAC and the internal PHY is provided by the internal MII. The LAN91C111 also supports the EXT-PHY mode for use of an external PHY. It provides a flexible slave

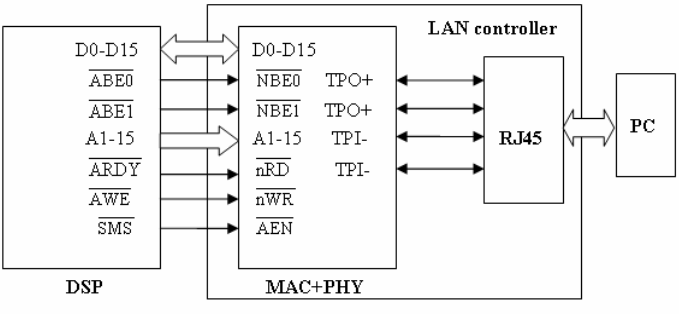


Fig. 6. Connection between DSP and LAN controller

interface for easy connectivity with industry standard buses. The Bus Interface Unit (BIU) can handle synchronous as well as asynchronous transfer, with different signals being used for each one. The total internal memory FIFO buffer size is 8k bytes. These properties are satisfied to the system request. The TJ45 adopt HFJ11-2450s chip. The system bus is 16-bit. So the $\overline{\text{ABE0-1}}$ pins of DSP connect to the $\overline{\text{NBE0-1}}$ pins of network chip. $\overline{\text{NBE2-3}}$ connects to high impedances. The hardware connection is shown in figure 6.

3.5 Other Peripheral Circuit Design

In order to provide real-time debugging for program in DSP, we connect JTAG with the emulator in our system. Furthermore, it also can be used to burn data and programs to FLASH. The periodical update task of face feature information is also completed by JTAG. [11-13] The UART interface is the equipment between DSP and the door manager controller, which can send message to open door or play warning sound.

The power equipment of system is TPS70302PWP. It can supply double adjustable voltage. One is 3.3V for peripheral circuits; the other is 1.2V for the kernel of DSP. To avoid destroying DSP by error of incorrect electrifying sequence, the system supplies electricity with peripheral circuits at first, and then DSP. The automatic and manual reset signal outputted by the power source do logical 'AND' operation before they are connected to DSP and peripheral circuits so as to reset system automatically or manually when the supplied voltage of system is too low. To prevent from the disturbance of high frequency caused by the high clock frequency of peripheral circuit, which can reduce the stability of system, peripheral clock circuit supplies 20MHz clock for DSP to match its kernel clock by PLL of DSP. [7]

4 Software Design

Firstly, the system will be initialized as soon as it is power on. Whenever someone passes the door, he or she can press the specified key to send interruption signal system. The system will mask the external interruption when it received the signal, and then it sends instructions to the image sensor to capture an image. The image sensor will notify DSP through interruption that the capture operation ends when it has completed to capture an image. Then the system starts to process the image and recognize. Finally it gives the recognition result and opens the door in so far as right or gives a warning message (the speaker is beeping) for wrong. Figure 7 shows the main flowchart of our system.

During the system initialization, DSP reads data from pm, mean, facetpl, eyetp1, and then store them in the variable of Pm, Mean, Facet, Eyet respectively for the following feature extraction to use. The data of captured image are stored in the fdata variable. The image sensor can capture the image without any software; it is different from the existing system in which the captured image should be converted through A/D before processing. The captured image of face will be performed unitary operation on size and

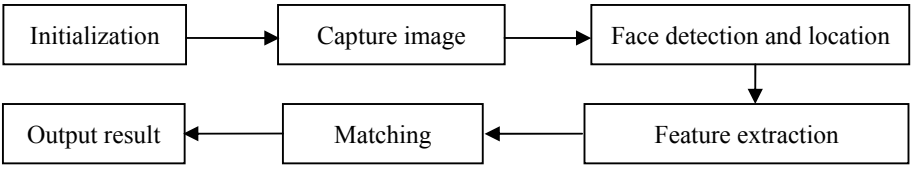


Fig. 7. System main flowchart

gray degree. The following step is to judge whether there is a face existed in the picture and give the accurate location if a face existed. Program locates such parts of the face as head, face, and eyes pupils. The flow chart is shown in figure 8.The method of detection and location in our system is based on the traditional template matching, and it is used for those images, which are static and gray, captured under simple background and even shine. The result will be better if the image is positive face picture.[14-15]The method of traditional template matching is to calculate the Euclidean distance between the captured image matrix and the face template(or eye template).

After the location, the feature extraction is beginning. Firstly the data of face location subtract the data of even face stored in Mean, and then an array can be generated, the array is projected at the feature face matrix pm to extract the feature value. The method of feature extraction and selection in our system is PCA (Principal Component Analysis). [15-17]

Then program matches the extracted feature value with those templates that have been trained and stored. The algorithm for matching is NNA (Nearest Neighbor Algorithm). [18-19] In addition, the face and eye templates for detection and location, the feature face template for extracting the feature value and the match template for matching are all trained and generated in PC beforehand and burned to the FLASH of our system. The match template is composed of the face feature value so that FLASH can accommodate more people's feature value in the limited space.

The maximum count of studying & training samples is 5 to 8 for every person. These samples have been appended in out system by generating image so as to improve the rate of recognition.[20]The sample is generated and append by a series of transformation. The main transformations are mirror transformations; rotate transformations of the picture and the changing of gray information of bottom for an image.

The most important aspect in the whole system is to select the threshold value of similar degree. The similar degree of our system is measured by cosine angle of two eigenvectors.[21] The threshold value determines the safety and efficiency of our system. If the threshold value is too high, those people who have the right to pass the door can't pass it; on the contrary, if the threshold value is too low, the reliability of system is reduced, as a result, those people who have not the right to pass the door can pass it. There are two factors to judge whether the similar degree threshold value is good or not. One is the mistake-passing rate; another is the mistake-refusing rate on the threshold value. Generally, both of factors are restricted mutually. If the mistake-passing rate were low, the mistake-refusing rate would be high. If the mistake-refusing rate were low, the mistake-

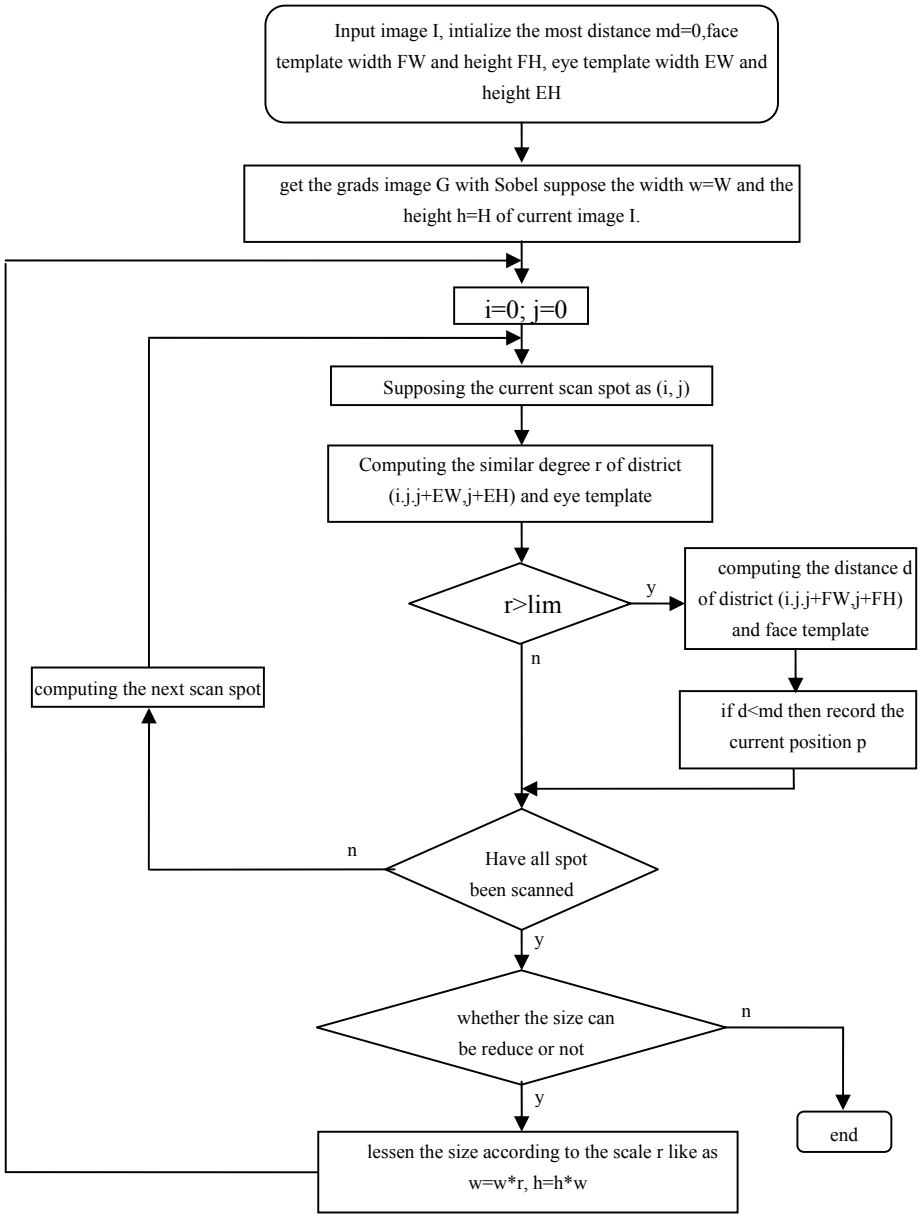


Fig. 8. Flowchart of detection and location

passing rate would be accordingly high. The curve in Figure 9 shows the variation of the mistake-passing rate and the mistake-refusing rate following the variation of the similar degree threshold value.

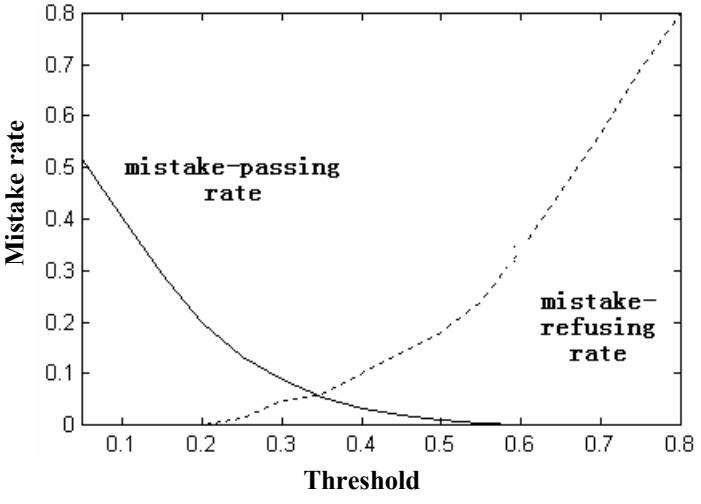


Fig. 9. Variation of the mistake-passing rate and the mistake-refusing rate following the variation of the similar degree threshold value

Figure 10 shows the results of our system.

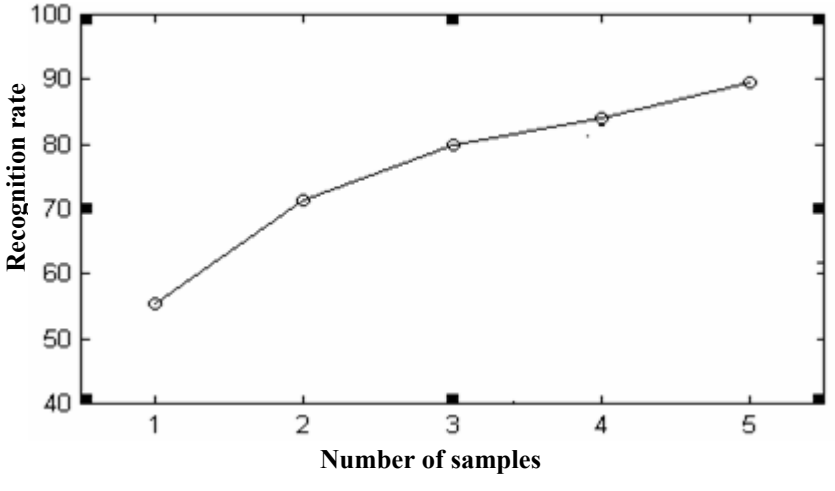


Fig. 10. Result of Recognition in system

5 Conclusion

In this paper, we have designed a hardware framework, which based on the face recognition of door manager requirement. DSP is the kernel of the door manager

system and face recognition algorithm has been applied in it. We have implemented the hardware interface circuit design of the system and the algorithm testing. The result is good under the still light. In the same time, the system integrates image sensor, DSP, FLASH, network chip and other peripheral equipments on a single board. Not only it is smaller in size, lower in cost, but also it can work faster and more reliably. In a word, it will be applied in many fields in the future.

Acknowledgement

This work was supported by the National Natural Science Foundation of China under Grant No.60433020, and the Key Science-Technology Project of the National Education Ministry of China under Grant No.02092.the Key Laboratory for Symbol Computation and Knowledge Engineering of the National Education Ministry of China.(93K-17).

References

1. Wang, C.M., Sun, H.B., (2004), System Design and Development Instance for DSP of TMS320C5000, Publishing House of Electronic Industry.
2. Zhang, X.W., Chen, L., (2003), Principle and Development Application of DSP, Publishing House of Electronic Industry.
3. Yu, M, Zhen, H., (Vol.19.No.6, 2003), Face Recognition System Based on DSP, Microcomputer Information Press, P: 55.
4. HDCS-1020/2020 CMOS Image Sensors Datasheet.
5. ADSP_BF531_2_3_a datasheet.
6. Zheng, J.F., (No.7, 2004), VHDL Design and Implement of I2C Bus Design of Electronic Application Press.
7. DSP Principle and System Design of Blackfin, P120-122.
8. ADSP-BF533 Blackfin Processor Hardware Reference, P197.
9. MT48LC4M16A2TG DATASHEET.
10. SMSC The LAN91C11 DATASHEET.
11. Xiao, Y.N., Yi, J., Song, Y.S., (No.4, 2004), DSP Boot under Emulator Environment for TMS320C6000, Overseas Electronic Component Press, P33.
12. Ding, J., Wei, F., Li, G.Y., (No.8, 2003), Research of FLASH Building in DSP for System Program and Boot Loading Method Collaterally, Electronic Technology Application Press, P78.
13. Visual DSP++4.0 User's Guide Datasheet.
14. Surendra Ranganath, Krishnamurthy Arun, (Vol.30.No.10, 1997), Face Recognition Using Transform Features And Neural Network, Pattern Recognition Press, P1615-1622.
15. Shao, Q., Zhang, XG., (2000), Pattern Recognition, Publishing house of Tsinghua University, P223-227.
16. Alex(Sandy) Pentland, Tanzeem Choudhury, (No.2, 2000), Face Recognition for Smart Environments, IEEE Computer, P50-55.

17. Ian Craw, Takashi Kato, Shigeru akamatsu, (Vol.21.No.8, 1999), How Should We Represent Faces for Automatic Recognition, IEEE Transaction on Pattern Analysis and Machine Intellignece.
18. Mark J.L.Orr, (No.8, 1996), Introduction to Radial Basis Function Nerworks.
19. Abhijit S.Pandya Robert B.Macy, Pattern Recognition and Implement in Neural Network, Publishing House of Electronic Industry.
20. Terence Sim, Rahul Sukthandar, Matthew Mullin, Shumeet Baluja, (2000), Memory-based Face Recognition for Visitor Identification, Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition.
21. Zhou, C.G., Cui, Z., Zhang, Y., Duan, J., (Vol.23, No.12, 2002), Visiting and Controlling System Based on Face Recognition, Microcomputer System Press.

AlchemistJ: A Framework for Self-adaptive Software*

Dongsun Kim and Sooyong Park

Department of Computer Science and Interdisciplinary
Program of Integrated Biotechnology, Sogang University,
Shinsu-Dong, Mapo-Gu, Seoul, 121-742, Republic of Korea
{darkrsw, sypark}@sogang.ac.kr

Abstract. The major goal of self-adaptive software is to provide a mechanism that allows a software system to dynamically change its architectural configuration during run-time to cope with requirement changes and unexpected conditions. Software which needs to handle dynamically changing internal and external environment is one of the areas in which self-adaptive software may do an important role in improving the reliability and performance of software systems. There are three main capabilities that are necessary to support self-adaptive software: the ability to monitor and recognize internal/external situations that affect behavior of the software system; the ability to determine when and what to reconfigure in the software system to handle the situations; and the ability to dynamically change the software architecture during run-time to make the reconfiguration effective. In this paper, we describe a software framework to support such capabilities to realize self-adaptive software and its experiment results.

1 Introduction

Intelligent software systems embedded in dynamic devices such as intelligent service robots should provide various capabilities in various environment continuously. This requirement makes the internal operations in software systems more complex. Due to the complexity, unpredictable errors and changes of situation often occurs at run-time[10]. Those unpredictable errors and changes of situations disturb maintaining proper performances of a system at run-time. In addition, it is a major factor which makes stable and persistent services of a system impossible.

In particular, the embedded software, which needed in more dynamic and complex changes of environment, faces the following problems:

1. *Various unexpected situations* that cannot be handled by pre-programmed actions

* This research was performed for the Intelligent Robotics Development Program, one of the 21st Century Frontier R&D Programs funded by the Ministry of Commerce, Industry and Energy of Korea.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 98–109, 2005.
© IFIP International Federation for Information Processing 2005

2. *Changing user requirements* that are encountered after deploying the embedded software
3. *Faults and errors* (in embedded hardware and software) that make some parts of embedded software unusable or malfunctioned
4. *Frequent replacement and reconfiguration* of hardware components, which may cause some inconsistency problems in software that depends on the hardware components

To overcome the problems above, the embedded software need to have the ability to maintain and improve its own capabilities. The embedded software should maintain three capabilities even in complex and exceptional situation of external environment, which are changing dynamically. The followings are three capabilities to maintain and improve its own functionalities:

1. *Interfaces* for implementing recognition of environment
2. *Decision making and learning* for the adaptive behavior by using reinforcement learning
3. *Dynamic reconfiguration* by using role- and port- based software architecture

Three capabilities above cannot directly overcome all situations such as unexpected problems, changes of user's requirements, disorders in hardware and software, and inconsistency in a system. However, the capabilities offer flexibility and opportunity for software to adapt itself against environmental changes. Self-adaptive software attaches new operations, modifies the current operations, and removes unnecessary operations in order to overcome problems. In this paper, we propose a framework for self-adaptive software called 'AlchemistJ'.

We explain the framework proposed in this research in detail in section 2, and then show a case study using Robocode in section 3. We comment on other significant work in Section 4 before drawing conclusions and proposing future research in section 5.

2 Approach

2.1 Overall Architecture

AlchemistJ consists of the Monitor package('env'), the Decision maker & Learner package('learner') and the Reconfigurator package('recon'). The Monitor package contains interfaces to support observation of situations and rewards of environment. Each situation is a vector of factors which can affect behavior of the software system. Each reward is a real-numbered value that evaluates behavior of the software system. The Decision maker & Learner package contains components to support decision making and learning of software. This makes decisions based on situations and rewards given by a monitor. The Reconfigurator package provides components to maintain software architecture of the systems and reconfigure the architecture based on adaptation decisions from the Decision Maker. AlchemistJ assumes software is designed by using role- and port-based software architecture and reconfigures software architecture at run-time.

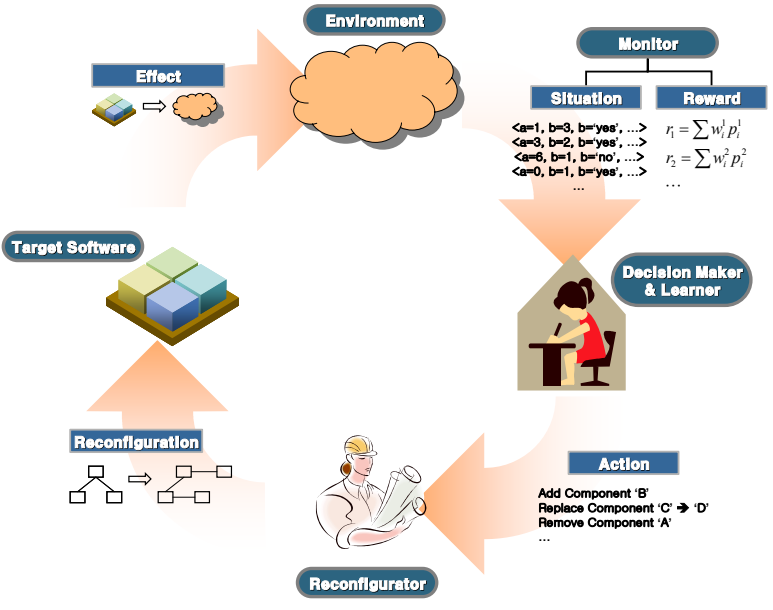


Fig. 1. The Process of AlchemistJ

Reconfigurations leads to changes of behavior of the software system. The changed behavior affects its environment. This affection changes situations of its environment because of not only the changed behavior but also other factors(for example, human factors, randomness, etc). The software system observes a new situation and reward by using the Monitor. The Learner updates its knowledge using the observed situation and reward. The software system repeats this iterations at run-time to adapt its operations and to accumulate experience.

Fig. 1 shows the process of AlchemistJ explained above.

2.2 The Monitor

The Monitor package defines basic interfaces for observation of environment and a process for passing situations and rewards to the Decision Maker and Learner. This package has three interfaces, 'IState', 'IReward' and 'IEnvironment'.

'IState' defines how to implement situation information. 'IState' has two methods as follows:

1. `public String getStateCode()`
2. `public void setStateCode(String code)`

'getStateCode()' returns the unique situation code which is a string of characters. 'setStateCode()' replaces the unique situation code with a given situation code. Developers should implement a monitor instance which is appropriate for

a target domain to observe environment and implement a class to contain situation information using 'IState'. Each object of the class contains a situation of environment. Each situation is mapped to the unique situation code. 'IReward' defines how to express reward information. 'IReward' has one method as follows:

```
1. public double getReward()
```

'getReward()' returns a real-numbered reward value. This value is determined by a predefined equation. This equation should be defined by developers. AlchemistJ basically assumes software maximize rewards. If a developer is eager to minimize rewards, the Monitor should be implemented to multiply rewards by '-1'.

'IEnvironment' defines connections of the Monitor, the Decision maker & Learner and the Reconfigurator. 'IEnvironment' provides four methods:

```
1. public IState getState()
2. public boolean giveAction(IAction action)
3. public IReward getReward()
4. public ActionSet getAdmissibleActionSet(IState s)
```

Developers should implement a component in compliance with 'IEnvironment'. 'getState()' returns the current situation of environment. The current situation is a object of the class which implements 'IState'. 'giveAction()' gives an adaptation decision made by the Decision Maker to the Reconfigurator. The Reconfigurator reorganizes components of the software system and composes them into a new architecture. 'getReward()' returns rewards which are observed from environment. Each reward is a instance of the class which implements 'IReward'. 'getAdmisibleActionSet()' returns a set of possible adaptation actions. These actions describe architecture-based adaptation actions such as component replacement, component attachment and topology changes. The Decision Maker determines one of the adaptation actions as the current adaptation decision.

2.3 The Decision Maker and Learner

AlchemistJ applies reinforcement learning to the Decision Making & Learning. Reinforcement learning is a mathematical formulation of interaction between an agent and its environment[13]. There are lots of specific algorithms that carry out reinforcement learning. AlchemistJ offers Q-learning implementation which is one of the specific algorithms of reinforcement learning[14][15]. Q-learning assumes a modeless environment. In other words, software cannot know deterministic model of state transitions.

The Decision Maker receives the current situation as input data from the Monitor. Then, the Decision Maker chooses an action to be applied. After applying the action to the system, The Learner receives the next situation and the immediate reward from the Monitor and updates Q-values, $Q(s, a)$. This updating is carried out by equation (1):

$$\begin{aligned} &Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a)], \\ &where \quad 0 < \alpha < 1, \quad 0 < \gamma < 1 \end{aligned} \tag{1}$$

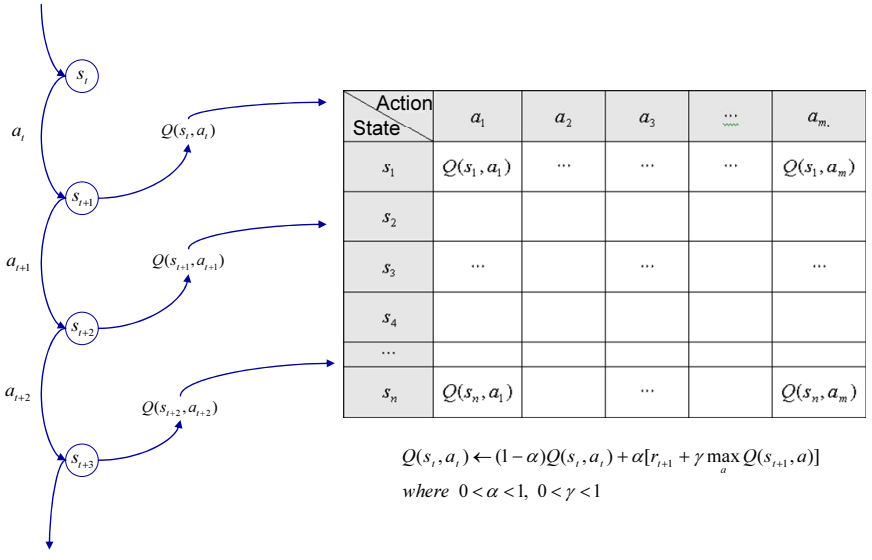


Fig. 2. Q-learning accumulates experiences of the software system by updating Q-values

In equation (1), $Q(s_t, a_t)$ indicates the Q-value for the situation s_t and the taken action a_t where t is time. The term $(1-\alpha)Q(s_t, a_t)$ represents knowledge that the software system experienced and the term $\alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a)]$ represents the current experience where r_{t+1} is an immediate reward from monitor and γ is a discount factor[13]. The value α controls weights of two terms.

Fig. 2 depicts a conceptual process of Q-learning in the Decision Maker & the Learner. In each state the Decision Maker takes an action and moves to the next state. In every transition, the Learner records rewards which is evaluated by the monitor.

'learner' package implements the Decision Maker & Learner. 'AbstractLearner' contains general functionalities for decision making & learning. This is connected with the Monitor which implements 'IEnvironment' interface. 'AbstractLearner' receives situation and reward information from the Monitor through 'IEnvironment' interface and makes decisions and learns its environment using the information. Basically, AlchemistJ offers 'QLearner' as an implementation of 'AbstractLearner'. If developers want to design a new decision maker & learner, they should implement a class that inherits 'AbstractLearner' and connect it with the Monitor.

2.4 Role- and Port-Based Software Architecture

To make self-adaptation possible, the software system should contain flexible components. These components should be reorganized into architecture to op-

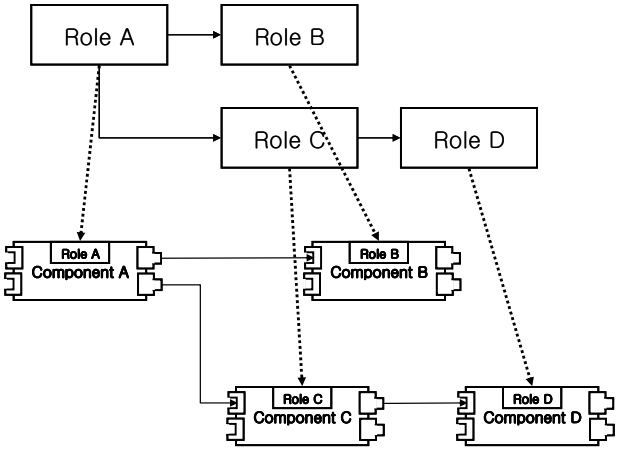


Fig. 3. Correspondence between roles and components

erate its functionalities. For this reason, self-adaptive software development requires predefined component & architecture design principle.

AlchemistJ provides role- and port-based architecture[9][6][3] as a design principle. Assuming software is designed by role- and port-based architecture design method, AlchemistJ reconfigures target software architecture. To construct role- and port-based architecture, developers must decompose software into roles. Each role describes abstract functionalities of each independent part of software. Roles do not express concrete capabilities or strategies. In other words, they should not define 'how to do', but 'what to do'. A role can be connected to make use of another role. After defining roles, they should be composed into a role-model. This role-model is a basis of software architecture.

A role-model describes relationship among roles. The Reconfigurator of AlchemistJ makes it possible to change a role-model at run-time. A role can be attached in and removed from the role-model. Each role should correspond to a component. This component should play the corresponding role. Each component consists of component body, role description, required ports, provided ports. The component body contains executable code that the component executes. The role description indicates a role that the component plays. One component can play only one role. The required ports describes other components(actually provided ports) that the component needs. The Provided ports describe what the component can do. One provided port should be connected with only one required port. Fig. 3 shows a sample of correspondence between roles and components.

2.5 Reconfigurator

AlchemistJ supports reconfiguration of roles and components at run-time. Role reconfiguration consists of role attachment, role removal and changes of topology.

Component reconfiguration comprises component attachment because of role attachment, component removal because of role removal, changes of a component topology because of changes of role topology and component replacement. These capabilities are implemented in 'recon' package.

'Reconfigurator' maintains the role-model and components in software architecture and reorganizes components into a new architecture. Components maintained by 'Reconfigurator' should comply role- and port-based architecture. To do this, 'recon' package offers five interfaces. 'IPort' contains basic methods for ports. 'IPortRequired' and 'IPortProvided' which inherit 'IPort' describe required ports and provided ports respectively. All components should implements 'IComponent' interface which provides basic operations of components. 'IRole' contains basic operations to describe a role. Every component has an object implementing 'IRole'. 'recon' package basically provides 'SimpleRole' as an implementation of 'IRole'.

3 Experiment

3.1 Introduction to Robocode

Robocode is a platform which includes the environment for game development and a simulator and is developed by IBM. The game simulator in Robocode provides a rectangular field for a battle. In the field, robots participating battle away each other in order to destroy the opposing robot. In order to simplify the implementation, this paper supposes that only two robots can participate in a battle.

3.2 Implementation of a Robot Using AlchemistJ

We have implemented the robot's capability to recognize environment according to interfaces provided by 'env' package in the framework. As mentioned in Section 2.2, 'env' package provides three kinds of interfaces. First, 'RobotState' class implements 'IState' interface. 'RobotState' class contains the number of hit by bullets and the number of hit by wall. This class converts and encodes the current situation into the unique situation code. 'RobotReward' class implements 'IReward' interface. An equation for rewards is set up as follows.

$$r_t = 0.3 \cdot \triangle E_S + (-1) \cdot 0.7 \cdot \triangle E_E \tag{2}$$

In equation (2), r_t, the reward value in time t, is calculated by summation of two terms, $\triangle E_S$ and $\triangle E_E$. $\triangle E_S$ is the difference of the robot's energy between time $t-1$ and t. $\triangle E_E$ is the difference of the enemy's energy between time $t-1$ and t. Weights of two terms are 0.3 and 0.7 respectively. Since the energy of the enemy should be minimized, we multiplied $\triangle E_E$ by (-1).

3.3 Construction of Software Architecture

We design a robot in compliance with Role- and port-based software architecture proposed in section 2.4. Fig. 4 shows the roles of the robot and the relationship among them.

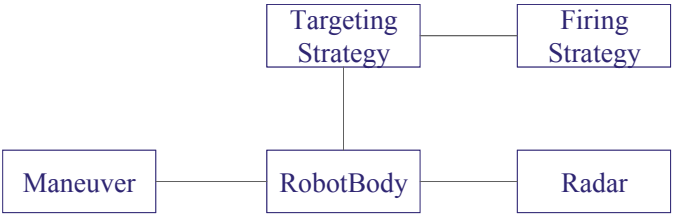


Fig. 4. The Role-model of the Robot

Table 1. The component list of robot architecture

Component name	Role of component	Required ports	Provided ports
NewRobot	RobotBody	RadarPort, ManeuverPort, TargetPort	N/A
Radar	Radar	N/A	RadarPortP
LinearManeuver	Maneuver	N/A	ManeuverPortP
CircularManeuver	Maneuver	N/A	ManeuverPortP
AdvancedTargeting	Targeting Strategy	FiringPort	TargetPortP
RammingTargeting	Targeting Strategy	FiringPort	TargetPortP
LinearTargeting	Targeting Strategy	FiringPort	TargetPortP
Firing	Firing Strategy	N/A	FiringPortP

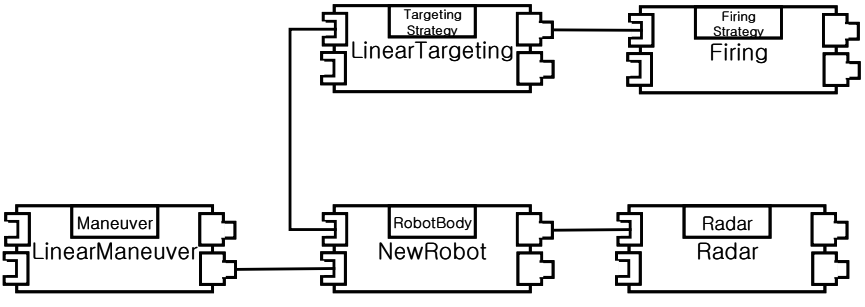


Fig. 5. The Initial Architecture of the Robot

The robot has five roles. 'Maneuver' is responsible for how to move. 'Radar' determines how to find an enemy robot by using a radar, and 'Targeting Strategy' determines how to target the enemy robot. 'Firing Strategy' determines how to fire at the enemy robot. 'RobotBody' has a role to relay over to the other 4 roles. In this experiment, 'Maneuver' and 'Targeting Strategy' are the target of adaptation.

Based on the components shown in table 1, the robot is constituted. Fig. 5 shows the structure of the components assigned roles of the robot in the initial

state. The 'LinearManeuver' component works a role as 'Maneuver' and the 'LinearTargeting' does as 'Targeting Strategy' in the initial state of the robot.

3.4 The Result from An Experiment

We have performed two experiments in order to verify effectiveness of adaptation of the robot implemented by AlchemistJ. The first experiment is a battle our robot which has no adaptation capabilities(Robot X) and an enemy robot from outside. We design our robot without adaptation capabilities but this robot has 'CircularManeuver' component which plays 'Maneuver' role and 'AdvancedTargeting' component which plays 'Targeting Strategy'. 'CircularManeuver' component outperforms 'LinearManeuver', and 'AdvancedTargeting' component outperforms 'RammingTargeting' and 'LinearTargeting'. The enemy of this robot is 'Antigravity Robot' which is one of the best robots. Fig. 6 shows that 'Antigravity Robot' has a predominant winning average against a robot without adaptation capabilities. On the basis of this experiment, we designed the next experiment.

The second experiment is a battle between our robot with adaptation capabilities(Robot A) and 'Antigravity Robot'. 'Robot A' can reconfigure its architecture using components which plays 'Maneuver' role and 'Targeting Strategy' role such as 'LinearManeuver' and 'RammingTargeting'. Fig. 7 shows the result from the battle between two robots.

The result from the battle between 'Robot A' and 'Antigravity Robot' shows that 'Robot A' has a 60∼70% winning average against 'Antigravity Robot'. It means that the dynamic reconfiguration with diverse 'Maneuver' and 'Targeting Strategy' components gives better performance than the maintenance with only one moving component and one targeting component even though those outperform other components.

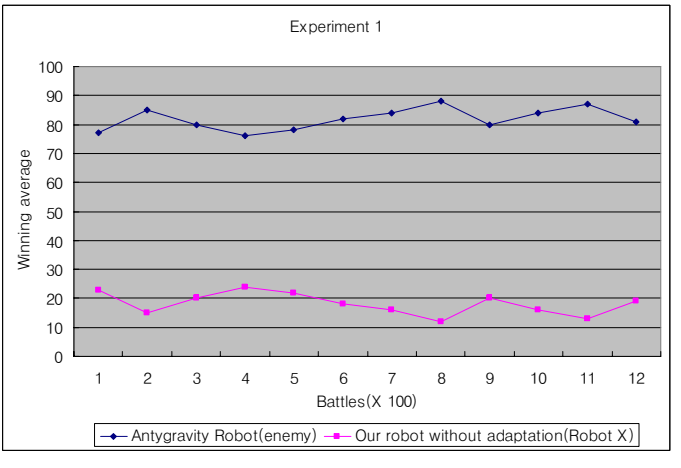


Fig. 6. The Result of the First Experiment

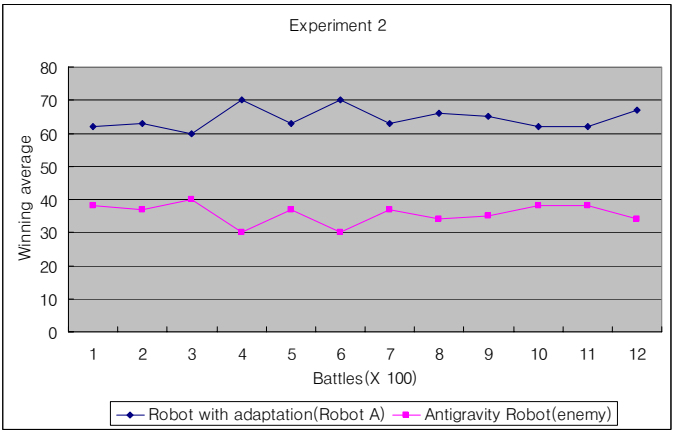


Fig. 7. The Result of the Second Experiment

Our approach has the exact difference from a sequence of `if-then-else` statements. A sequence of `if-then-else` statements is hard-coded in source code. Each `if-then-else` statement consists of a condition and an action. To modify a condition or an action, a programmer must rewrite the source code, compile it, shutdown the software system to be modified, re-link the compiled component, and re-execute again. However, AlchemistJ provides the characteristics to reduce the burden of a programmer. Of course, the programmer can make a situation(condition)-behavior(action) lookup table and propose a mechanism for updating the table dynamically, but the mechanism is already another version of AlchemistJ.

4 Related Work

The researches of Richard N. Taylor(University of California, Irvine)[12][11][2], David Garlan(Carnegie Mellon University)[4][1][5], Jamie Hillman(Lancaster University)[8][7] provide the capability of dynamic reconfiguration. The capabilities of dynamic reconfiguration proposed in each framework are different, but they have similarity in terms of changing software structure in the architecture level. All of the three frameworks cannot provide decision making and learning of the adaptive behavior. Even though they mention its necessity, they cannot suggest what technology and theory to use and to implement in practice. Except Garlan's research[5], they do not provide a implementation, but conceptually mention only the necessity of it.

We proposed the AlchemistJ framework integrating three capabilities. AlchemistJ is implemented by Java, and comprises three packages to support recognition of environment by defining interfaces for the monitor, decision making and learning by Q-learning implementation, and dynamic reconfiguration using role- and port-based software architecture.

5 Conclusions

Self-adaptive software recognizes environments, and determines the adaptation behavior according to the situations it recognized. In addition, it can change its own architecture in order to perform the adaptation behavior.

We proposed AlchemistJ as a software framework to support developing self-adaptive software. This framework proposed in this paper supports the followings.

1. It prescribes interfaces to support the implementation for recognition of its environment.
2. It employs reinforcement learning to the framework in order to support decision making and learning of the adaptive behavior.
3. It prescribes role- and port-based software architecture and reconfiguration principle in order to support dynamic reconfiguration at run-time.

In order to verify operations of the framework, we have implemented an instance of self-adaptive robot software with Robocode, and examined the ability of adaptation of software from two experiments.

Further study is the development of visualization tools and measurement tools to improve the efficiency of the framework. Also we need to develop a repository for framework to maintain components. In AlchemistJ, we implemented a simple and local repository which can only gives a component corresponding a simple message. The more advanced repository should manage components with more specific description and give components via online connection.

References

1. R. J. Allen, R. Douence, and D. Garlan. Specifying dynamism in software architectures. In *Proceedings of the Workshop on Foundations of Component-Based Software Engineering*, 1997.
2. E. M. Dashofy, A. van der Hoek, and R. N. Taylor. Towards architecture-based self-healing systems. In *Proceedings of the First Workshop on Self-Healing Systems*, 2002.
3. K. R. Dixon, T. Q. Pham, and P. K. Khosla. Port-based adaptable agent architecture. In *First International Workshop, IWSAS 2000*, 2000.
4. D. Garlan, S.-W. Cheng, A. C. Huang, B. Schmerl, and P. Steenkiste. Rainbow: Architecture-based self-adaptation with reusable infrastructure. *IEEE Computer*, 37(10):46–54, October 2004.
5. D. Garlan, B. Schmerl, and J. Chang. Using gauges for architecture-based monitoring and adaptation. In *the Working Conference on Complex and Dynamic Systems Architecture*, 2001.
6. M. M. Gorlick and R. R. Razouk. Using weaves for software construction and analysis. In *Proceedings of the 13th International Conference on Software Engineering*, 1991.
7. J. Hillman and I. Warren. Meta-adaptation in autonomic systems. In *10th IEEE International Workshop on Future Trends of Distributed Computing Systems*, 2004.

8. J. Hillman and I. Warren. An open framework for dynamic reconfiguration. In *26th International Conference on Software Engineering*, 2004.
9. T.-H. Kim and Y.-G. Shin. Role-based decomposition for improving concurrency in distributed object-oriented software development environments. In *23rd International Computer Software and Applications Conference*, 1999.
10. R. Laddaga. Active software. In H. S. Pual Robertson and R. Laddaga, editors, *First International Workshop on Self-Adaptive Software(IWSAS 2000)*, volume 1936 of *Lecture Notes in Computer Science*, pages 11–26. Springer, 2000.
11. P. Oreizy. Issues in the runtime modification of software architectures. Technical report, Department of Information and Computer Science, University of California, Irvine, 1996.
12. P. Oreizy, M. M. Gorlick, R. N. Taylor, D. Heimbingner, G. Johnson, N. Medvidovid, A. Quilici, D. S. Rosenblum, and A. L. Wolf. An architecture-based approach to self-adaptive software. *IEEE Intelligent Systems*, 14(3):54–62, May 1999.
13. R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, Massachusetts, The MIT Press, 1998.
14. C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, 1989.
15. C. J. C. H. Watkins and P. Dayan. Technical note: Q-learning. *Machine Learning*, 8(3-4):279 – 292, May 1992.

Design and Implementation of Accounting System for Information Appliances

Midori Sugaya[1], Shuichi Oikawa[2], and Tatsuo Nakajima[1]

[1] Department of Computer Science, Waseda University
[2] Department of Computer Science, University of Tsukuba
{doly, tatsuo}@dcl.info.waseda.ac.jp, shui@cs.tsukuba.ac.jp

Abstract. This paper presents the design and implementation of Accounting System that is a resource monitoring and restriction system. The system improves the reliability and security of a system. Accounting System is a generic to offer various services, such as security improvement, overload control, class-based accounting, and resource reservation that require CPU resource control.

1 Introduction

Information appliances [5] are important elements to realize the ubiquitous computing vision [3]. Nowadays, most consumer electronics appliances have computing capability in order to retrieve data from sensors, to process the data, and to control devices. The recent emergence of information appliances requires more advanced features, such as networking and GUI. Those features dramatically complicate the appliances, software systems and increase their code sizes.

Networked systems need to be prepared for security attacks through the Internet. Since we expect users to be system administrators of applications, their software systems must be more robust than personal computers. Software bugs can also cause the monopolization of CPU resources. Real-time operating systems may be vulnerable to bugs of real-time programs that easily consume the whole CPU resources. For example, a problem can happen when multimedia applications process continuous media streams. Those multimedia applications have strict timing constrains, and are given the real-time priority in general. Therefore, once they process the streams, they can easily monopolize the whole CPU resources. In such a situation, the GUI process that is usually executed on the time-sharing scheduler cannot consume necessary CPU capacity, so that a user cannot control the multimedia applications through the GUI buttons. Another example is an overload condition. If an overload condition occurs, the response time of the system becomes worse.

General embedded operating systems have not yet provided a resource protection mechanism that aims to protect the CPU resources from such monopolizations and overload situations. Therefore, it is necessary to offer a generic mechanism to restrict the use of CPU resources to develop reliable information

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 110–120, 2005.
© IFIP International Federation for Information Processing 2005

appliances. In this paper, we propose Accounting System, a general-purpose resource monitoring and restriction system that prevents the excessive use of the CPU capacity of a process or a group of processes.

The following two points are our design principles.

- *Simple design* to be applied to various services. Accounting System focuses on providing a simple and generic model and interface. Therefore, the system should be able to be easily applied to various kinds of services.
- *Accurate resource management* by using a fine-grain resolution timer. Future information appliances require to support a fine-grained rate based execution that can be realized by the accurate resource management, for offering better response time and more stable execution.

Information appliances have become very complex. Most of them nowadays contain web browsers, Java Virtual Machines, and many other applications. Implementing these applications in a robust way needs a more powerful operating system. Linux is an open-source operating system, and supports various CPU architectures. Those features are very suitable for information appliances. Therefore, many industries consider adopting Linux for their products. While the use of Linux is increasing, it does not restrict the resource consumption for their processes. We assume Accounting System can solve this problem, and implement it in the Linux kernel.

The remaining of the paper is structured as follows. Section 2 describes the design and model of Accounting System. Section 3 presents the architecture and implementation. Section 4 shows the services that can be implemented by Accounting System, such as the secure resource protection, class-based accounting, and overload monitoring. Section 5 presents the evaluation results. Finally, in Section 6, we conclude the paper.

2 Accounting System Model

Accounting System and Accounting Object: Accounting System is a system to offer the functions that monitor and restrict the execution time of a process or a group of processes. We have developed it as a part of the kernel services. Accounting System provides an abstraction to manipulate the CPU capacity of each process named *accounting object*. An accounting object represents a capacity of the CPU resource on a single processor. The CPU time of each process bound to an accounting object is accumulated in it. Through an accounting object, a user can control the allocation of CPU resources for each accounting object. When a user binds a process to an accounting object through the system call, the process is given the restricted execution time specified by its accounting object.

Time Management Model: To realize the restriction, an accounting object has two parameters C and T, where T represents a period that is a constant period to control the object, and C is the maximum time to be able to execute processes within T. The process bound to the accounting object cannot consume

the CPU time more than C within T. When processes consume the entire CPU time within each period T, the process is blocked until the next period comes.

3 Implementation

3.1 Callback Hooks

In Accounting System, callback hooks are introduced into the process management system in the host kernel for the purpose of catching events about processes. Those hooks catch the relevant events, and deliver those events to Accounting System. On our design principle, Accounting System should be independent of the host kernel, and the only minimum points in the host kernel should be modified. Accounting System assumes that the four callback hooks as described next are inserted in the host kernel.

- *schedule_hook:* It catches scheduling events on context switching among processes. The hook delivers the scheduling events to Accounting System that changes the state of an accounting object.
- *fork_hook:* It catches events when a new process is created. These events are required to start the accounting for the newly created processes. If a parent process has already been bound to an accounting object, its child processes inherited the same accounting object.
- *exit_hook:* It catches events on process termination. The events notify the timing to unbind the termination of a process from its accounting object.
- *kernel_callback_hook:* It catches events when a system call returns from the kernel mode to the user mode. Accounting System checks the state of an accounting object for the blocking of the process to execute the system call.

3.2 Timer Management System

The management of timers has an important role of Accounting System. In Accounting System, there are two types of timers, a replenishment timer and an enforcement timer. A replenishment timer is a periodic timer for each accounting object that is constantly expired. An enforcement timer keeps to watch on the elapsed time of the currently executing process. They are implemented by using kernel software timers. In the following paragraph, we describe the details of them.

- *Replenishment timer:* It manages the periodic time of an accounting object. Each accounting object has its own replenishment timer. When an accounting object is created, the timer is also created. In the replenishment timer, first the expiration time is set by adding the periodic time to the current time. If the elapsed time reaches the expiration time, the registered timer handler is executed to set the next expiration time. These replenishments are recursively done by the replenishment timer until it is destroyed.

– *Enforcement timer:* It watches the elapsed time of the currently executing process. The timer accumulates the execution time of the process, until it reaches to the maximum execution time of the accounting object. At that time, the registered timer handler is invoked to change the state of the accounting object. By using schedule_hook, the enforce timer catches the events about context switching among processes, and stops and starts its timer to monitor the CPU usage of processes.

3.3 Account Object Management Operations

To manage accounting objects, the system provides the accounting object management function. It introduces the following accounting object's basic operations. To receive the accounting object services, an application firstly invokes the *create* operation to allocate the accounting object in the kernel memory. Each created accounting object has a unique object ID. To bind a process to the specified accounting object with the object ID, the application invokes the *bind* operation. When the application no longer requires the accounting services, it invokes the *destroy* operation to terminate the accounting object service and to free the object memory in the kernel space. The *set* operation changes the parameters of an accounting object. The *get* operation obtains the parameter in an accounting object.

3.4 State Management

There are three states to represent an accounting object's status. *null* status represents the status that an accounting object is created, but no process is bound to the accounting object. *running* status represents the accounting object is running, and the status also represents that available time are left within the period time. *depleted* status represents the status that no more available time is left within the period time. The *depleted* status also represents that the accumulated execution time of a process is reached to the maximum execution time specified in the accounting object.

The state of an accounting object is changed by events. At first, when a process binds to a specific accounting object, the state becomes *running*. If the sum of the execution time of the bound processes equals to the maximum execution time of the accounting object, the object changes its state to *depleted*. Accounting System here to take action to stop their execution or deliver the notice to the bound process which is on the excessive use of the CPU capacity. The actions are classified into *block* and *signal*. If it is *block*, the system puts the processes to the wait queue. When the next expiration time comes, the replenish timer checks the wait queue, and if it finds the processes in the wait queue, sets the *running* state to the accounting object, and wakes up the processes. When a process uses excessive CPU capacity, a signal is delivered to the process. Using the state transition mechanism, Accounting System restricts the CPU usage of the processes accurately.

3.5 High Resolution Timer

For Accounting System, the high-precision clock and timestamp counter are essential to maintain the fidelity of its accuracy. To provide less than 10ms resolution for monitoring, there are some approaches. The most straightforward way is to make the clock interrupt frequency higher than 100Hz in order to issue more interrupts. This allows the kernel to provide a precise timing and to increase the system responsiveness. However, shorter intervals require CPU to spend a longer time in the kernel mode. Therefore, user programs run slower because of the system overhead to handle interrupts. Our system solves the problem by using the high resolution timer [2]. The high-resolution timer in Intel x86, is implemented by using the one shot mode of the APIC clock timer chip. Therefore, the extra timer interrupt overhead will never happen. The high-resolution timer allows us to specify microsecond granularity parameters in accounting objects.

4 Services Using Accounting System

4.1 Secure Resource Protection

Accounting System protects the CPU resource from downloaded programs that behave maliciously. We assume that there is a manager process that starts processes to execute downloaded programs. The manager process is bound to an accounting object, of which C and T are 50ms and 100ms, respectively. When the manager process receives a request to execute a downloaded application, it creates a new process. The process is automatically bound to the accounting object to which the manager process is bound. Therefore, the newly created process cannot consume more than 50% of CPU capacity, and it cannot monopolize the CPU resource.

Binding Multiple processes: Accounting System provides the *bind* operation that binds a process to the accounting object. We also provide some useful interfaces to bind multiple processes to an accounting object. For example, if a malicious program consecutively invokes the *fork* system call, the whole system resources may be consumed. We implement the following two methods. The one is to inherit an accounting object when creating a child process. The other is to bind an accounting object using group ID. The former function is realized by using a hook within the *fork* system call. If a parent process is bound to an accounting object, the child process who is forked from the parent inherits the same accounting object. Thus, the same restriction as its parent is applied. The latter uses UNIX group process ID. The system binds the group of processes to a specific accounting object. It is safer than binding all processes to the same accounting object respectively. For example, when several commands connected by pipes are started, they can be easily bound to the same accounting object by using group ID.

Access Control: Each accounting object has an owner attribute for controlling the access to a specific accounting object. The owner attribute of the accounting object is assigned by a process that creates it. For the consideration of the security, our system allows them to be manipulated only by their owners except the privileged user. In resources reservation systems such as Linux/RK [4], there is no support of access control. Therefore, these previous system cannot be used for protecting the CPU resource from malicious programs.

Kernel Interface: The following is the kernel interface offerd by Account System.

- *account_create(&object_id,&object_attr)* It creates a new accounting object. When created, the object is not bound to any processes. A privileged or owner application program sets the parameters, such as C, T, in the object_attr memebers.
- *account_destroy(&object_id)* It destroys the accounting object specified by object_id. This deletes the associated timers, and frees the memory of the accounting object in the kernel address space.
- *account_bind/unbind_pid(object_id, pid_t)* It binds/unbinds the specified accounting object through object_id to a process whose process ID is pid. An accounting object can be bound to multiple processes. If the process is terminated, the process is automatically unbound from the accounting object.
- *account_get/set(object_id, &object_attr)* It retrieves/changes the parameters of the specified accounting object.

4.2 Class-Based Accounting

In this section, we propose a class-based accounting as the one of the services that uses Accounting System. A class is defined here as a group of processes that belong to a specific scheduling class, such as the real-time scheduling class. The class-based accounting is realized by binding the processes belonging to the same scheduling class.

A problem of process starvation: When a user forgets to bind a real-time process to an accounting object, the real-time process may monopolize the entire CPU resource. Even if most real-time processes are restricted, the only one real-time process can consume the rest of all the CPU resources; thus, all other time-sharing processes are starved. To avoid such a situation, the system designer should estimate the total maximum utilization of the accounting objects, which bind to the all real-time processes. We called the approach class-based accounting.

Class-based Accounting: The class-based accounting is realized by binding processes in the specific scheduling class to an accounting object. A class contains all processes that are scheduled by the same scheduling policy. The system allocates the CPU resources proportionally to each scheduling class. The total proportions of the classes are set to 100%. For example, a user can allocate 40%

of the CPU capacity for the real-time scheduling class, and the remaining 60% of the CPU capacity for the time-sharing scheduling class. The system assumes to have two classes that are necessary to allocate the absolute rate of the CPU capacity.

The processes bound to each class cannot excessively use the maximum utilization of the CPU resources that are set in each scheduling class. Since the real-time process cannot use more than the assigned capacity, the time-sharing process can use the remaining capacity without process starvation.

Implementation: The current class-based accounting allocates the CPU capacity for the real-time and time-sharing scheduling class. The accounting object for each scheduling class is created at the boot time. When it is created, all processes that belong to the same scheduling class are bound to the same accounting object. In order to set the parameter easily, class-based accounting adopts weight as its interface parameters. The weight is the parameter that represents a proportion and can easily be specified like (2,3). If weight (2,3) given, the real-time class and the time-sharing class are allocated 40% and 60% CPU capacity respectively. In class-based accounting, the accounting object of each scheduling class should use the same period time. Also, a user needs to set the resolution time that is a period for the accounting objects. For example, if the resolution is 100ms and the class proportion is 40%, the class is received 40ms within the 100ms period.

Kernel Interface: We design the kernel interface for class-based accounting as described as following.

– *weight_set (wt_ts, wt_rt, res)* Sets the weight and resolution to the accounting object for respective scheduling classes. When the class accounting object is started, it monitors the execution time of the bound processes.

4.3 Overload Monitoring

Conceiving the overload situation: For general-purpose operating systems, controlling overload situations is very important to offer services in a stable way. In such critical situations, the computational requests from the process will exceed the time that is available in the system. As for time-sharing processes, the average response time is increasing. We propose the monitoring function that keeps to watch the total utilization of the CPU. If the overload condition happens, the overload monitoring system issues a signal to notify to the registered administrator process. If there is a notification for the administrator process, some recovery procedures can be taken to stabilize the system.

Design and Implementation: The overload monitoring can also be realized by Accounting System. In this system, an accounting object bound to the idle process is used to monitor the total utilization of the system resources. The idle process is executed when there is no runnable process. If the system becomes busy, the idle process consumes a very little CPU capacity. Through monitoring

the CPU usage of the idle processes, the system can detect the overload situation. Our system allows an application to receive a signal when the CPU usage of the idle process becomes lower than the specified CPU. The service is available when a system designer creates an overload accounting object, and sets the parameters to it. There is the only one overload accounting object in a system. A system designer should explicitly set the C and T parameter to the overload accounting object. Suppose the designer sets C and T to 75ms and 100ms respectively, it monitors the CPU capacity not to exceed 75%. If the designer would like to check the overload condition every minute, C and T should be set to 45sec and 60sec respectively. If the total usage of CPU utilization exceeds 75%, an overload notification signal is delivered to a specific process. A system designer also should set a process as a receiver of the signal. The signal number can be freely decided by a user. In consideration of the security, only a privileged user can create and destroy the overload accounting object.

Kernel Interface: The overload monitoring offers the kernel interface as described as follows.

- *overload_create (&object_attr, pid)* The function binds an idle process to the overload accounting object. The user sets an administrator process ID that accepts a signal from the system and a signal number that is used to deliver the signal to the administrator process when an overload condition occurs.
- *overload_destroy(void)* It destroys the overload accounting object.

4.4 Resource Reservation

An application running on our system can request the reservation of a certain amount of CPU resources. The OS-enforced resource reservation can be introduced similar to CPU capacity reserves [1] [4], which provide the framework for managing the processor capacity. Accounting System can implement the framework easily by adding a mechanism for the admission control as a user-level service.

5 Evaluation

We have evaluated Accounting System by running several benchmarks. The evaluation uses a standard PC that has Celeron 300MHz CPU and 512MB of RAM, and the results were measured using the built-in high-resolution timestamp counter.

5.1 Accuracy of Accounting System

This section evaluates the accuracy of Accounting System. To evaluate the effectiveness of high-resolution timer support, we use the benchmark program that executes the infinite-loop. We create two accounting objects that set 20ms and

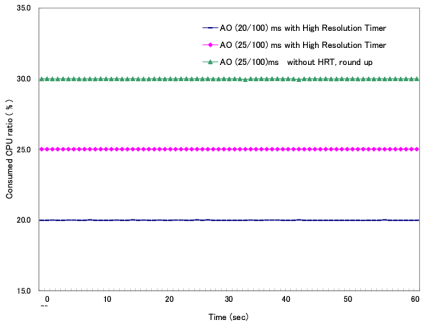


Fig. 1. With/Without High Resolution Timer Support

25ms of CPU time every 100ms, and bind the program to each accounting object. Figure 1 shows the result of the consumed CPU ratio with/without the support of a high-resolution timer. In this figure, the bottom two lines are the results of the high-resolution timer support. It shows that each 20% and 25% resource restriction is correctly limited. In the contrast, without the high-resolution timer support, the program bound to the accounting object whose execution time is 25ms executes 30ms within 100ms. These are apparently due to the fact that Linux updates its internal timer-clock at the 10ms intervals. Even if a user sets a parameter whose resolution is less than 10ms, it is impossible to ensure the resolution. The results shows that the effectiveness of the accurate resource accounting.

5.2 Effectiveness of CPU Protection

This section evaluates the effectiveness of the CPU resource protection in Accounting System. First, we show the effect of the security attacks, then, show the result of the class-based accounting.

Attacks to create multiple processes: Figure 2 shows the consumed CPU ratio of multiple processes bound to an accounting object. To evaluate the effectiveness for protecting from attacks to create multiple processes, we run an evil

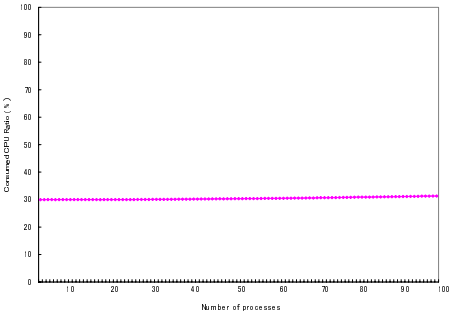


Fig. 2. The Numbers of Process and Accounting Object Ratio

program that invokes fork() consecutively and produces a lot of child processes that execute their infinite-loops. The response time becomes very bad by this malicious program because these processes consume the whole CPU resource. In the evaluation, if the first process is bound to the 30% restricted accounting object, the forked processes cannot use more than 30% of the CPU capacity, even if the numbers of processes are increasing. The result shows the effectiveness of Accounting System as shown in Figure 2. The loss of the accuracy appears as the numbers of created process is increased. In the current implementation, a process that executes a system call is blocked after the system call is returned, when the execution time of the accounting object is expired. If there are multiple processes that are bound to the same accounting object, and they execute system calls, the system calls can be executed before blocking themselves. This is a reason to exceed the specified execution time when the number of process bound to the same accounting object is increased.

Class-based Accounting: Figure 3 and Figure 4 show the results of the evaluation of class-based accounting. To evaluate the effectiveness of the class-based accounting, the benchmark runs a real-time process to execute an infinite loop with/without the class-based accounting. Figure 3 shows the result when the class-based accounting is not used. The result shows that all CPU resources are consumed by the real-time process. Then, we evaluate the class-based accounting in which each class is set the weight (4,1) where 80% and 20% CPU capacity are allocated for real-time processes and time-sharing processes respectively. The Xserver (X Window System display server) process runs on the time-sharing scheduler. On Xserver, some game programs are running. Figure 4 shows the result that the real-time processes can not exceed 80% CPU resources, where the Xserver process can use 20% CPU resource at a maximum.

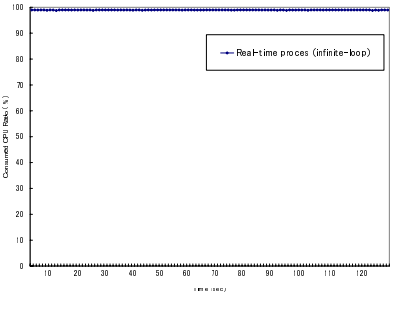


Fig. 3. Without Class-Based Accounting

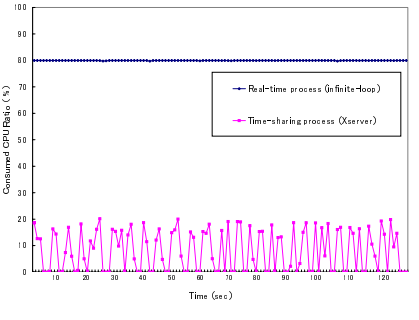


Fig. 4. With Class-Based Accounting

6 Conclusion

In this paper, we have proposed Accounting System. This system provides the generic accounting model that can be applied various services that are used

to increase the system security and reliability. It also supports a fine-grained resource accounting with the high-resolution timer. This makes the system more stable because the application can obtain the necessary rate precisely even in the microsecond resolution. The performance evaluation results have showed that the system effectively protects the malicious use of CPU resources.

References

1. Clifford W. Mercer, Stefan Savage, and Hideyuki Tokuda.: Processor capacity reserves: Operating system support for multimedia applications. In Proceedings of the IEEE Internation Conference on Multimedia Computing and Systems, pages 90–99, May 1994.
2. High Resolution Timer, http://high-res-timers.sourceforge.net/
3. Mark Weiser.: The Computer for the 21st Centriy, Scientific American, Vol 265, No.3, 1991.
4. Shuichi Oikawa and Ragunathan Rajkumar.: Portable RK: A portable resource kernel for guaranteed and enforced timing behavior. In Proc. IEEE RTAS, 1999.
5. W. P. Sharpe, S. P. Stenton.: Information Appliances, The Human-Computer Interaction Handbook : Fundamentals, Evolving Technologies and Emerging Applications - Human Factors and Ergonomics, Chapter37, Lawrence Erlbaum Assoc Inc Published 2002.

Loop Distribution and Fusion with Timing and Code Size Optimization for Embedded DSPs*

Meilin Liu[1], Qingfeng Zhuge[1], Zili Shao[2], Chun Xue[1],
Meikang Qiu[1], and Edwin H.-M. Sha[1]

[1] University of Texas at Dallas,
Richardson, Texas 75083, USA
{mxl024100, qfzhuge, cxx016000, mxq012100, edsha}@utdallas.edu
[2] Hong Kong Polytechnic University,
Hung Hom, Kowloon, Hong Kong
cszlshao@comp.polyu.edu.hk

Abstract. Loop distribution and loop fusion are two effective loop transformation techniques to optimize the execution of the programs in DSP applications. In this paper, we propose a new technique combining loop distribution with direct loop fusion, which will improve the timing performance without jeopardizing the code size. We first develop the loop distribution theorems that state the legality conditions of loop distribution for multi-level nested loops. We show that if the summation of the edge weights of the dependence cycle satisfies a certain condition, then the statements involved in the dependence cycle can be distributed; otherwise, they should be put in the same loop after loop distribution. Then, we propose the technique of maximum loop distribution with direct loop fusion. The experimental results show that the execution time of the transformed loops by our technique is reduced 21.0% on average compared to the original loops and the code size of the transformed loops is reduced 7.0% on average compared to the original loops.

Keywords: Loop Fusion, Loop Distribution, Embedded DSP, Code Size, Scheduling.

1 Introduction

Timing performance and code size are two major concerns for embedded systems with very limited on-chip memory resources [9]. Since loops are prevalent in multimedia processing, digital signal processing, etc., loop transformations such as loop fusion, loop distribution are needed to optimize the execution of the loops in embedded DSP applications [8, 1, 2, 3, 5].

Loop distribution separates independent statements inside a single loop (or loop nest) into multiple loops (or loop nests) [8, 1, 3, 2, 5]. Loop distribution

* This work is partially supported by TI University Program, NSF EIA-0103709, Texas ARP 009741-0028-2001, NSF CCF-0309461, NSF IIS-0513669, Microsoft, USA.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 121–130, 2005.
© IFIP International Federation for Information Processing 2005

can be used to break up a large loop that doesn't fit into the cache [2, 3, 8, 1]. It can also improve memory locality by fissioning a loop that refers to many different arrays into several loops, each of which refers to only a few arrays. Loop distribution can also enable other loop optimization techniques such as loop fusion, loop interchanging and loop permutation [8, 1, 3, 5].

Loop fusion, on the other hand, groups multiple loops to increase the instruction level parallelism, and correspondingly reduces execution time. There are a lot of previous works using loop fusion to optimize the execution of loops [2, 3, 8, 1, 4, 5, 7]. But sometimes loop shifting or retiming is needed to enable loop fusion, which will cause the code size expansion because of the generation of the prologue and epilogue. The maximum loop fusion technique (Max_LF) proposed in [4] can maximize the opportunities of loop fusion, but it cannot be applied in the cases where strict memory constraint applies.

A combination of loop distribution and loop fusion can find an optimization solution with both reduced execution time and restricted code size if the loop properties are fully understood. Therefore, it is very important to revisit and formalize the loop distribution and loop fusion theorems on graph models, and find an effective way to combine these two oppositely directed optimization techniques. In this paper, we propose a technique of *maximum loop distribution with direct fusion* (MLD_DF), which performs *maximum loop distribution*, followed by *direct loop fusion*. The technique significantly improves the timing performance compared to the original loops without jeopardizing the code size.

Loop distribution is an important part of our technique of maximum loop distribution with direct fusion (MLD_DF). But loop distribution is not simple. All the data dependences have to be preserved when breaking one single loop into multiple small loops. The authors of [2, 3, 5] stated that loop distribution preserves dependences if all statements involved in a data dependence cycle in the original loop are placed in the same loop. We show that dependence cycle is a restriction for loop distribution for one-level loops only. For multi-level nested loops, dependence cycle is not always a restriction for loop distribution. If the summation of the edge weights of the dependence cycle satisfies a certain condition, then the statements involved in the dependence cycle can be distributed.

In this paper, we propose general loop distribution theorems for multi-level loops to state the legality conditions of loop distribution based on the understanding of loop properties on graph models. Then, we show how to conduct maximum loop distribution for N-level loops based on the loop distribution theorems. We then propose the technique of maximum loop distribution with direct fusion (MLD_DF). The experimental results showed that the execution time of the transformed loops by our MLD_DF technique can be improved 21.0% on average compared to the original loops when there are eight functional units.

The rest of the paper is organized as follows: We introduce the basic concepts and principles related to our technique in Section 2. In Section 3, we propose the loop distribution theorems to guide loop distribution. We propose the technique of maximum loop distribution with direct fusion (MLD_DF) in Section 4. Section 5 presents the experimental results. Section 6 concludes the paper.

2 Basic Concepts

In this section, we provide an overview of the basic concepts and principles related to our technique.

2.1 Data Flow Graph

We use a multi-dimensional data flow graph($MDFG$) to model the body of one nested loop. A MDFG $G = (V, E, \boldsymbol{d}, t)$ is a node-weighted and edge-weighted directed graph, where V is the set of computation nodes, $E \subseteq V \times V$ is the set of edges representing dependences, $\boldsymbol{d}$ is a function from E to Z^n, representing the multi-dimensional delays between two nodes, where n is the number of dimensions, and t is a function from V to positive integers, representing the computation time of each node.

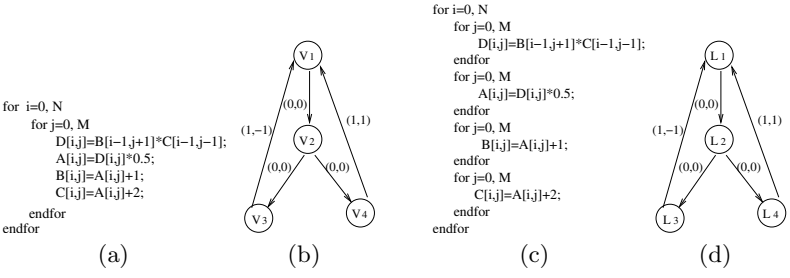


Fig. 1. (a) A loop extracted from a Wave Digital Filter. (b) The corresponding Data Flow Graph. (c) The distributed loop. (d) The loop dependence graph of the distributed loop.

The execution of each node in V exactly once represents an iteration, i.e., the execution of one instance of the loop body. Iterations are identified by a vector $\boldsymbol{i}$, equivalent to a multi-dimensional index. Inter-iteration dependences are represented by vector-weighted edges. For any iteration $\boldsymbol{j}$, an edge e from u to v with delay vector $\boldsymbol{d}(e)$ means that the computation of node v at iteration $\boldsymbol{j}$ requires the data produced by node u at iteration $\boldsymbol{j} - \boldsymbol{d}(e)$. An edge with delay $(0, \cdots, 0)$ represents a data dependence within the same iteration. A legal MDFG must have no zero-delay cycle, i.e., the summation of the edge weights along any cycle can not be $(0, \cdots, 0)$.

The program shown in Fig. 1(a) is extracted from a wave digital filter and its corresponding data flow graph is shown in Fig. 1(b).

2.2 Loop Dependence Graph

Loop dependence graph (LDG) is a higher-level graph model compared to the data flow graph [4]. It is used to model the data dependences between multiple loops. A multi-dimensional loop dependence graph (MLDG) $G = (V, E, \delta, o)$

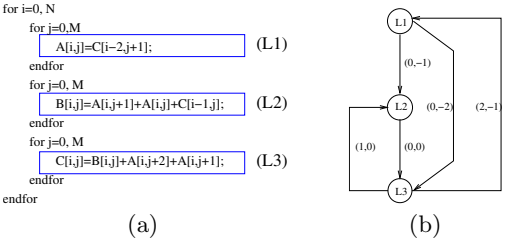


Fig. 2. A loop and its corresponding LDG

is a node-labeled and edge-weighted directed graph, where V is a set of nodes representing the loops. $E \subseteq V \times V$ is a set of edges representing data dependences between the loops. δ is a function from E to Z^n, representing the minimum data dependence vector between the computations of two loops. o is a function from V to positive integers, representing the order of the execution sequence. All the comparisons between two data dependence vectors are based on the lexicographic order in this paper.

The loop dependence graph of the loop in Fig. 2(a) is shown in Fig. 2(b). There are three nodes $V = \{L1, L2, L3\}$ in the loop dependence graph that represent the three innermost loops in the program. The loop dependence edges are $E = \{e_1 : L1 \rightarrow L2, e_2 : L1 \rightarrow L3, e_3 : L2 \rightarrow L3, e_4 : L3 \rightarrow L2, e_5 : L3 \rightarrow L1\}$. The data dependence vectors are $\{(0,-1),(0,0)\}$ between nodes $L1$ and $L2$, $\{(0,-2),(0,-1)\}$ between nodes $L1$ and $L3$, $\{(0,0)\}$ between nodes $L2$ and $L3$, $\{(1,0)\}$ between nodes $L3$ and $L2$, and $\{(2,-1)\}$ between nodes $L3$ and $L1$. According to our MLDG definition, $\delta(e_1) = (0,-1)$, $\delta(e_2) = (0,-2)$, $\delta(e_3) = (0,0)$, $\delta(e_4) = (1,0)$, $\delta(e_5) = (2,-1)$.

In a loop dependence graph, a fusion-preventing dependence is represented by an edge e with edge weight $\delta(e) < (0,0,\cdots,0)$. The fusion-preventing dependence edges for the LDG shown in Fig. 2(b) are e_1 and e_2.

A backward edge in the loop dependence graph is defined as an edge from a node labeled with a larger number to a node labeled with a smaller number. For example, in the loop dependence graph shown in Fig. 2(b), node $L1$ represents the first inner loop of the loop shown in Fig. 2(a), which is labeled with 1 according to the execution sequence. Node $L3$ represents the third inner loop of the loop shown in Fig. 2(a), which is labeled with 3. According to the definition, in the loop dependence graph shown in Fig. 2(b), the backward edges include the edge from node $L3$ to $L1$, and the edge from $L3$ to $L2$.

3 Theorems of Loop Distribution

In the process of loop distribution, we must maintain all the data dependences to ensure that we won't change the semantics of the original program. To guarantee the correctness of loop distribution, we propose several theorems to guide loop distribution. In this paper, we only consider the loops with uniform data

dependences. A lot of DSP applications fit in this category. In the following, the general theorems for the N-level nested loops is proposed.

Theorem 1. *Given a N-level perfect nested loop and its corresponding data flow graph, after loop distribution, if the shared outer loop level is J, $J \leq N-1$, then the backward edge e in the LDG $G = (V, E, \delta, o)$ of the distributed loop must satisfy that $(\delta_1(e), \cdots, \delta_J(e)) \geq (0, \cdots, 1)$.*

If we distribute the loop on the $(J+1)$-th loop level, then the distributed loop has J-level shared outer loop. A backward edge e in a LDG represents loop-carried data dependence. The first J elements in an edge weight vector $(\delta_1(e), \cdots, \delta_J(e))$ represent the dependence distance of the shared outer loop. If $(\delta_1(e), \cdots, \delta_J(e))$ of a backward edge e in the LDG of the distributed loop is a non-positive value, then true data dependences are changed to anti-data dependences by loop distribution. Thus, loop distribution becomes illegal. Obviously, the code shown in Fig.3(b) computes differently from the code shown in Fig.3(a). $A[i, j]$ is dependent on $B[i, j-1]$, which is a true data dependence in the original program as shown in Fig.3(a). If we directly distribute the loop shown in Fig.3(a), then this true data dependence becomes an anti-data dependence in the distributed loop as shown in Fig. 3(b). Therefore, any legal LDG of an N-level nested loop (distributed loop) must have a positive value on the first J elements of the weight vectors of the backward edges.

```
for  i=0, N
     for   j=0, M
           A[i,j]=B[i,j−1]*2;
           B[i,j]=A[i,j]+5;
     endfor
endfor
```

(a)

```
for  i=0, N
     for   j=0, M
           A[i,j]=B[i,j−1]*2;
     endfor
     for   j=0, M
           B[i,j]=A[i,j]+5;
     endfor
endfor
```

(b)

Fig. 3. An illustration example

Fig. 1(c) shows the distributed loop of the original program shown in Fig. 1(a). Because the backward edges $e_1 : L3 \rightarrow L1$ and $e_2 : L4 \rightarrow L1$ in the LDG shown in Fig. 1(d) have the edge weight (1,-1) and weight (1,1) respectively, i.e., the backward edges e_1 and e_2 both have positive value on the first element, the correct execution of the original loop is able to be preserved in the distributed loop.

When there are dependence cycles existing in the data flow graph of a loop, it's important to know whether the computations involved in a dependence cycle can be distributed or not. In the following, we show that the nodes in the dependence cycles of the LDG of a N-level nested loop can be *completely distributed* when the necessary condition in Theorem 2 is satisfied. A loop is completely distributed when each loop unit after distribution only has one array assignment statement.

Theorem 2. *Given a N-level perfect nested loop and its corresponding data flow graph* $G = (V, E, \boldsymbol{d}, t)$*,*

1. *if there is no dependence cycle in the data flow graph G,*
2. *or if for any dependence cycle* $c = \{v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_n \rightarrow v_1\}$ *in G, the summation* $\boldsymbol{d}(c)$ *of the edge weights of cycle c satisfies that* $(\boldsymbol{d}_1(c), \cdots, \boldsymbol{d}_J(c)) \geq (0, \cdots, 1), J \leq N - 1,$

then the loop can be completely distributed on the $(J + 1)$*-th level loop until the N-th level.*

Due to the space limit, we do not give a formal proof here. It directly follows from Theorem 1, since all the data dependences in the LDG of the distributed loop come from the data dependences of the original loop. Theorem 2 also shows that if we can distribute the loop at the J-th ($J < N$) loop level for a N-level loop, we can distribute this N-level loop at the innermost loop level.

For example, there is one cycle in the data flow graph of the 3-level loop shown in Fig. 4(a). The summation of the edge weights of the dependence cycle c in the corresponding data flow graph has the property that $d_1(c) = 1$. According to theorem 2, we can distribute the loop at the second loop level. The distributed loop is shown in Fig. 4(b). There is one-level shared outer loop in the distributed loop. Also we can distribute the innermost loop of this 3-level loop, and the distributed loop is shown in Fig. 4(c).

```
for  i=0, N
     for  j=0, M
          for  k=0, S
               A[i,j,k]=C[i−1,j,k]+7;
               B[i,j,k]=A[i,j,k]*3;
               C[i,j,k]=B[i,j,k]+5;
          endfor
     endfor
endfor
```

(a)

```
for  i=0, N
     for  j=0, M
          for  k=0, S
               A[i,j,k]=C[i−1,j,k]+7;
          endfor
     endfor
     for  j=0, M
          for  k=0, S
               B[i,j,k]=A[i,j,k]*3;
          endfor
     endfor
     for  j=0, M
          for  k=0, S
               C[i,j,k]=B[i,j,k]+5;
          endfor
     endfor
endfor
```

(b)

```
for  i=0, N
     for  j=0, M
          for  k=0, S
               A[i,j,k]=C[i−1,j,k]+7;
          endfor
          for  k=0, S
               B[i,j,k]=A[i,j,k]*3;
          endfor
          for  k=0, S
               C[i,j,k]=B[i,j,k]+5;
          endfor
     endfor
endfor
```

(c)

Fig. 4. (a)A 3-level nested loop. (b) The distributed loop with 1-level outer shared loop. (c) The distributed loop with 2-level outer shared loop.

Theorem 3 identifies the dependence cycle in the data flow graph that prevents the statements involved in the dependence cycle from distribution.

Theorem 3. *Given a N-level nested loop and its corresponding data flow graph* $G = (V, E, \boldsymbol{d}, t)$*, for any dependence cycle* $c = \{v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_n \rightarrow v_1\}$ *in G, such that the summation of the edge weights* $\boldsymbol{d}(c)$ *satisfies that* $(\boldsymbol{d}_1(c), \cdots, \boldsymbol{d}_J(c)) = (0, \cdots, 0)$*, the statements involved in dependence cycle c must be placed in the same loop after the loop is distributed on the* $(J + 1)$*-th loop level.*

If we distribute the statements in a dependence cycle c with $(\boldsymbol{d}_1(c), \cdots, \boldsymbol{d}_J(c)) = (0, \cdots, 0)$, then there will be a dependence cycle c with $(\boldsymbol{d}_1(c), \cdots, \boldsymbol{d}_J(c)) = (0, \cdots, 0)$ in the correspondent LDG of the distributed loop. For a legal program, all the edges e involved in a dependence cycle c in the LDG must have the property $(\boldsymbol{d}_1(c), \cdots, \boldsymbol{d}_J(c)) \geq (0, \cdots, 0)$. So the backward edge e' in the cycle c must have $(\boldsymbol{d}_1(e'), \cdots, \boldsymbol{d}_J(e')) = (0, \cdots, 0)$, which is contradictory to Theorem 1. When the first J elements of the summation of the edge weights of a cycle are all zeros, it indicates that all the statements in the cycle have to be computed within one loop. In other words, all the statements involved in a dependence cycle c with $(\boldsymbol{d}_1(c), \cdots, \boldsymbol{d}_J(c)) = (0, \cdots, 0)$ must be put into one loop after the loop is distributed on the $(J+1)$-th loop level.

4 Loop Distribution and Loop Fusion

In this section, we first use an example to show how to conduct maximum loop distribution for N-level nested loops based on the loop distribution theorems proposed in Section 3. Then, we propose the technique of maximum loop distribution with direct fusion (MLD_DF).

4.1 Maximum Loop Distribution

To conduct maximum loop distribution for a multi-level loop on the innermost loop level, we first remove the edges e with weight $(\boldsymbol{d}_1(e), \cdots, \boldsymbol{d}_{N-1}(e)) \geq (0, \cdots, 1)$ from the LDG of the given loop. Thus, if the summation $\boldsymbol{d}(c)$ of the edge weights of a cycle c in the LDG satisfies that $(\boldsymbol{d}_1(c), \cdots, \boldsymbol{d}_{N-1}(e)) \geq (0, \cdots, 1)$, then cycle c is broken. Then, we merge each cycle c with $(\boldsymbol{d}_1(c), \cdots, \boldsymbol{d}_{N-1}(c)) = (0, \cdots, 0)$ into one node. This is used to guarantee that the statements involved in the dependency cycle whose summation of the edge weights satisfies that $(\boldsymbol{d}_1(c), \cdots, \boldsymbol{d}_{N-1}(c)) = (0, \cdots, 0)$ will be put into the same loop after loop distribution. Then, we can reorder the nodes by the topological order to ensure that the edge weight $\delta(e)$ of a backward edge e in the LDG of the distributed loop has positive value on its first $N-1$ elements, i.e., $(\delta_1(e), \cdots, \delta_{N-1}(e)) \geq (0, \cdots, 1)$. Every node in the transformed graph corresponds to a loop unit in the distributed loop.

```
for  i=0, N
     for  j=0, M
          for  k=0, S
               A[i,j,k]=D[i,j,k−2];
               B[i,j,k]=A[i−1,j,k];
               C[i,j,k]=B[i,j,k]+D[i,j,k−1];
               D[i,j,k]=C[i,j,k]+7;
          endfor
     endfor
endfor
```

(a)

```
for  i=0, N
     for  j=0, M
          for  k=0, S
               B[i,j,k]=A[i−1,j,k];
          endfor
          for  k=0, S
               C[i,j,k]=B[i,j,k]+D[i,j,k−1];
               D[i,j,k]=C[i,j,k]+7;
          endfor
          for  k=0, S
               A[i,j,k]=D[i,j,k−2];
          endfor
     endfor
endfor
```

(b)

Fig. 5. (a)An example loop. (b) The distributed loop.

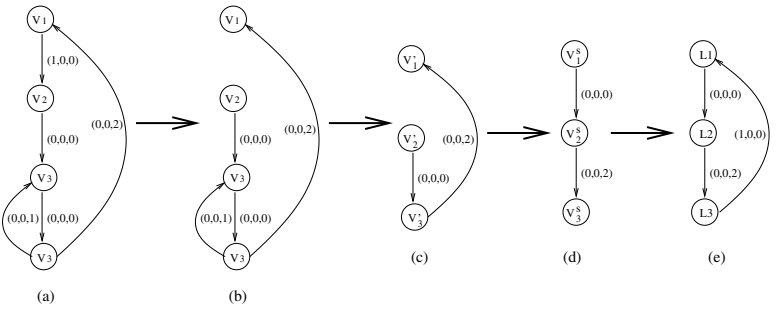


Fig. 6. The graph transformation process of the maximum loop distribution algorithm

We use a 3-level nested loop as shown in Fig. 5(a) to show the graph transformation process of maximum loop distribution for N-level nested loops. There are two cycles c_1, c_2 in its corresponding data flow graph as shown in Fig. 6(a). The summation of the edge weights of the first cycle $c_1 = \{V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_4 \rightarrow V_1\}$ satisfies that $\boldsymbol{d}(c_1) = (1, 0, 2)$, so cycle c_1 will be broken after the edges e with $(\boldsymbol{d}_1(e), \boldsymbol{d}_2(e)) \geq (0, 1)$ are removed. The summation of the edge weights of the second cycle $c_2 = \{V_3 \rightarrow V_4 \rightarrow V_3\}$ has the property that $\boldsymbol{d}(c_2) = (0, 0, 1)$, so cycle c_2 will be merged into one node according to the basic idea of maximum loop distribution since $(\boldsymbol{d}_1(c), \boldsymbol{d}_2(c)) = (0, 0)$. Thus, we get a DAG G'. Then, we perform topological sort on graph G' and obtain the node-reordered graph G^s shown in Fig. 6(d). Each node in the graph G^s corresponds to one loop unit in the distributed loop. According to the sorted nodes, we can generate the code of the distributed loop as shown in Fig. 5(b). Then we can get the LDG of the distributed loop as shown in Fig. 6(e). Fig. 6 shows the graph transformation process.

4.2 Maximum Loop Distribution with Direct Loop Fusion

Direct loop fusion is to fuse two or more loops into one loop when there are no fusion-preventing dependences between these loops. Our direct loop fusion technique is based on the loop dependence graph, and it is mapped to the graph partitioning technique [3, 5]. To apply direct loop fusion, we partition the loop nodes in the LDG into several partitions so that loop nodes connected by a fusion-preventing dependence edge are partitioned into different partitions. Thus, we guarantee that there is no fusion-preventing dependence existing between the nodes inside one partition and all the loop nodes inside one partition can be directly fused.

The technique of maximum loop distribution with direct fusion (MLD_DF) is to perform maximum loop distribution followed with direct loop fusion, such that the timing performance of the loops is increased without the increase of the code size. We first apply maximum loop distribution on a given loop. After we perform maximum loop distribution, we partition the loop nodes in the LDG of the distributed loop such that there is no fusion-preventing dependences existing

between the nodes inside one partition and all the loop nodes inside one partition can be directly fused [3]. Then, direct loop fusion is applied.

5 Experiments

This section presents the experimental results of our technique. We simulate a DSP processor with eight functional units. We compare the code sizes and the execution time of the original loops with those of the fused loops produced by the technique of maximum loop distribution with direct fusion (MLD_DF). The execution time is defined to be the schedule length times the total iterations. The schedule length is the number of time units to finish one iteration of the loop body. For the sequentially executed loops, the execution time is the summation of the execution time of each individual loop. The standard list scheduling algorithm is used in our experiments.

LDG1, LDG2, and LDG3 refer to the examples presented in Figure 2, 8, and 17 in [6]. LDG4 and LDG5 refer to the examples shown in Figure 2(a) and Figure 6(a) in [4]. Each node of an LDG is a DSP benchmark. The DSP benchmarks include WDF (Wave Digital filter), IIR (Infinite Impulse Response filter), DPCM (Differential Pulse-Code Modulation device), and 2D (Two Dimensional filter). We estimate the code size in memory by the number of instructions.

In Table 1, the column "*Original*" contains three fields: the number of loop nodes (#Node), the code size (Size), and the execution time (Time). The Column "*MLD_DF*" contains five fields: the number of loop nodes (#Node), the code size (Size), the execution time (Time), the improvement of the execution time (Time Impro.), and the reduction of the code size (Size Red.). The Column "*Original*" shows the number of the loop units, the code size and the execution time of the original loop. The Column "*MLD_DF*" shows the number of the loop units, the code size, the execution time, the improvement of the execution time of the fused loop by the MLD_DF technique and the reduction of the code size of the fused loop by the MLD_DF technique. Here, N is the total number of the iterations for the outermost loop, and M is the total number of the iterations for the innermost loop.

Although the maximum loop fusion technique (Max_LF) proposed in [4] can always achieve a shorter execution time than the technique of maximum loop dis-

Table 1. The code size and the execution time of the original loops and the transformed loops by the MLD_DF technique

	Original			MLD_DF				
Program	**#Node**	**Size**	**Time**	**#Node**	**Size**	**Time**	**Time Impro.**	**Size Red.**
LDG1	4	15	9*N*M	4	15	9*N*M	0%	0%
LDG2	7	84	36*N*M	2	74	26*N*M	27.8%	11.9%
LDG3	7	102	38*N*M	3	94	29*N*M	23.7%	7.8%
LDG4	3	36	23*N*M	2	34	17*N*M	26.1%	5.6%
LDG5	4	42	29*N*M	2	38	21*N*M	27.6%	9.5%
Improvement	-			-			21.0%	7.0%

tribution with direct fusion (MLD_DF), it increases the code size. In many cases, this technique cannot be applied because of the memory constraint. Compared to the Max_LF technique, the MLD_DF technique takes advantage of both loop distribution and loop fusion, so it reduces the original execution time and also avoids the code-size expansion. The experimental results showed that the timing performance of the fused loop by our MLD_DF technique can be improved 21.0% on average compared to the original loops, and the code size is reduced 7.0% on average compared to the original loops.

6 Conclusion

The maximum loop fusion technique (Max_LF) proposed in [4] can maximize the opportunities of loop fusion, but it also increases code size. In this paper, we developed the technique of combining loop distribution and loop fusion to achieve a shorter execution time with a reduction of the code size. The experimental results showed that the timing performance of the transformed loops by our MLD_DF technique can be improved significantly compared to the original loops, and the code size of the transformed loops is also reduced compared to the original loops.

References

1. R. Allen and K. Kennedy. *Optimizing Compilers for Modern Architectures: A Dependence-based Approach*. Morgan Kaufmann, 2001.
2. K. Kennedy and K. S. Mckinley. Loop distribution with arbitrary control flow. In *Proc. of the 1990 conference on Supercomputing*, pages 407 – 416, Nov. 1990.
3. K. Kennedy and K. S. Mckinley. Maximizing loop parallelism and improving data locality via loop fusion and distribution. In *Languages and Compilers for Parallel Computing, Lecture Notes in Computer Science, Number 768*, pages 301–320, 1993.
4. M. Liu, Q. Zhuge, Z. Shao, and E. H.-M. Sha. General loop fusion technique for nested loops considering timing and code size. In *Proc. ACM/IEEE International Conference on Compilers, Architectures, and Synthesis for Embedded Systems (CASES 2004)*, pages 190–201, Sep. 2004.
5. K. S. McKinley, S. Carr, and C.-W. Tseng. Improving data locality with loop transformations. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 18(4):424 – 453, July 1996.
6. E. H.-M. Sha, T. W. O'Neil, and N. L. Passos. Efficient polynomial-time nested loop fusion with full parallelism. *International Journal of Computers and Their Applications*, 10(1):9–24, Mar. 2003.
7. S. Verdoolaege, M. Bruynooghe, and F. Catthoor. Multi-dimensional incremental loop fusion for data locality. In *Proc. of the Application-Specific Systems, Architectures, and Processors*, pages 14–24, 2003.
8. M. Wolfe. *High Performance Compilers for Parallel Computing*. Addison-Wesley Publishing Company, Inc., 1996.
9. Q. Zhuge, B. Xiao, and E.-M. Sha. Code size reduction technique and implementation for software-pipelined DSP applications. *ACM Transactions on Embedded Computing Systems(TECS)*, 2(4):590–613, Nov. 2003.

Ensuring Real-Time Performance Guarantees in Dynamically Reconfigurable Embedded Systems*

Aleksandra Tešanović[1], Mehdi Amirijoo[1], Daniel Nilsson[1], Henrik Norin[1], and Jörgen Hansson[1,2]

[1] Linköping University, Department of Computer Science, Linköping, Sweden
{alete, meham, jorha}@ida.liu.se
[2] Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, USA
hansson@sei.cmu.edu

Abstract. In this paper we present a quality of service (QoS) adaptive framework for dynamic reconfiguration of component-based real-time systems. Our framework is light-weighted enabling reconfiguration in resource constrained embedded environments. Furthermore, it is possible to reconfigure both components and aspects of a system, hence, enabling finer tuning of a real-time system. Real-time QoS guarantees are maintained in the system and under reconfiguration by employing feedback-based scheduling methods.

1 Introduction

A large majority of computational activities in modern society are performed within embedded and real-time systems. For a real-time system, it is essential that results produced by the system are both produced correctly and in a timely manner. To ensure timeliness, tasks[1] in a real-time system are associated with deadlines, and a number of real-time scheduling techniques have been developed ensuring that tasks meet their respective deadlines (see [1]).

Successful deployment of real-time and embedded systems strongly depends on low development costs, a short time to market, and a high degree of reconfigurability. One way to meet these requirements is to adopt the component-based software development (CBSD) paradigm for real-time systems. This way systems are developed out of pre-defined software components to fit a specific real-time application. Component-based real-time system approaches, e.g., [2, 3, 4], do not provide efficient support for crosscutting features such as concurrency control or scheduling algorithms that typically crosscut the structure of the overall system. Aspect-oriented software development (AOSD) [5] has emerged as a new principle for software development that provides an efficient way of modularizing crosscutting concerns in software systems in "modules", called aspects. Using

* This work is supported by the Swedish Foundation for Strategic Research (SSF), the Swedish National Graduate School in Computer Science (CUGS), and the Center for Industrial Information Technology (CENIIT) under contract 01.07.

[1] Real-time systems are typically constructed out of concurrent programs called tasks.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 131–141, 2005.
© IFIP International Federation for Information Processing 2005

AOSD, systems can be built to contain only the required functionality, while other functional and non-functional crosscutting features encapsulated into aspects can be added to the system in a process called aspect weaving.

In an effort to integrate the two software engineering techniques, namely AOSD and CBSD, into real-time system development we have developed an approach to aspectual component-based real-time system development, ACCORD [6]. Given some system requirements ACCORD enables efficient system configuration from components and aspects from a library.

Most real-time component-based software systems, including our earlier work on ACCORD, are pre-compiled. This implies that the resulting running system is monolithic and not dynamically reconfigurable. Consequently, when updating or maintaining these systems, they need to be shut down for recompilation. However, reconfiguring a system on-line is desirable for embedded real-time systems that require continuous hardware and software upgrades in response to technological advancements, environmental change, or alteration of system goals during system operation [7, 3]. For example, small embedded real-time sensor-based control systems must be designed and developed such that software resources, e.g., controllers and device drivers, change on the fly. Hence, a reconfiguration mechanism, enabling adding removing, and exchanging components on-line, is needed to ensure that the software is updated without interrupting the execution of the system.

However, dynamic reconfiguration of a real-time system changes the temporal behavior of the system. For example, exchanging a component for a new version could result in a task execution time that is higher than the execution time obtained when using the existing component in the system. This in turn influences the real-time performance of the system negatively. Hence, the dynamic reconfiguration mechanism should be capable of ensuring that the system satisfies real-time performance guarantees, in terms of quality of service (QoS).

In this paper we present a QoS-adaptive framework for dynamic reconfiguration of real-time systems. The QoS-adaptive framework is an extension of the previous work on ACCORD to support exchange of components and aspects during run-time while preserving real-time performance guarantees. The main benefits introduced by this framework are as follows.

- The framework enables dynamic reconfiguration of real-time systems in embedded environments. We have identified a list of requirements that dynamically reconfigurable real-time and embedded systems need to satisfy, and we show that our framework fulfills these requirements.
- The framework is novel in that it supports dynamic reconfiguration in two dimensions: reconfiguration of components and reconfiguration of aspects. Dynamic reconfiguration of both components and aspects provides higher reusability and flexibility of a real-time system.
- The framework ensures that QoS guarantees are maintained even under reconfiguration. By employing feedback control structure we ensure that a real-time system adapts to the changes in the temporal behavior of components which occur during dynamic reconfiguration.

The paper is organized as follows. We formulate the problem in section 2. Our QoS-adaptive framework for dynamic reconfiguration is presented in section 3. The paper finishes with main conclusions in section 4.

2 Problem Formulation

In this section we identify a number of requirements that the reconfiguration framework needs to fulfill to ensure dynamic reconfigurability in a real-time system and discuss to what degree these requirements are fulfilled by existing approaches.

R1 A reconfiguration framework of a real-time system should be light-weight, implying that in normal operation of the system this framework must not introduce any significant overhead in the task execution and memory consumption of the system. This is to ensure that the system is usable in an embedded and real-time environment with sparse recourses in terms of CPU and memory.

R2 There should not be a restriction on the number of components that could be exchanged or added/removed from the system since reconfiguration may affect the entire system.

R3 Reconfiguration may be requested at any time, and the system has no a priori knowledge of the possible components that are to be reconfigured in the system.

R4 Reconfiguration must be carried out efficiently in terms of the granularity of exchangeable parts, implying that flexibility for fine-tuning of an embedded real-time application should be ensured. One should be able to exchange both functionality that is encapsulated into components and real-time algorithms that typically crosscut several components.

Accurate temporal characteristics of software components that are being added, removed, or exchanged in the system are not always available. Moreover, if temporal characteristics are available, the problem of determining exact temporal characteristics of software on a target platform is known to be difficult due to, e.g., cache and branch prediction [8]. Therefore, we need to ensure that, when reconfiguring the real-time system, the reconfiguration does not affect the performance of the system negatively. That is, a real-time system under reconfiguration should consider the varying temporal behavior of the software components being reconfigured and adapt accordingly. The adaptation should be such that specified performance requirements expressed in terms of a desired QoS, bounded worst-case QoS, and timely adaptation are satisfied. Hence, the following additional requirements on dynamically reconfigurable system arise.

R5 The system user must be able to specify the desired system QoS during steady state, i.e., the state in which no reconfiguration is applied.

R6 In the face of a transient state during which reconfiguration is applied, the worst-case system QoS and temporal adaptation must comply with the

specified requirements. More specifically, the QoS must satisfy worst-case system QoS requirements and the QoS must converge towards and reach the desired QoS within a certain specified time interval (also known as settling time).

The majority of approaches enabling development of configurable real-time systems is concerned with configuration of a real-time system in the design and compilation phase and does not provide flexibility by dynamically reconfiguring the system [9, 4, 3]. Existing approaches [10, 11, 2] enabling dynamic reconfiguration do not fulfill all of the identified requirements. The approach presented in [11] does not satisfy requirement R1, and is not suitable for embedded environments. Approaches that do comply with R1, e.g., [10, 2], do not provide flexible way of reconfiguration, e.g., system reconfiguration is done by pre-compiling possible configurations into the monolithic system, violating requirement R3 [2]. The named approaches to dynamic real-time reconfiguration do not provide means for exchanging algorithms and other crosscutting concerns in the system, and, therefore, are not fulfilling requirement R4. There has been work on maintaining performance guarantees under reconfiguration of real-time systems, addressing requirements R5 and R6, but this has been done assuming that the worst-case temporal behavior of components is known prior to the reconfiguration and/or that the reconfiguration is done off-line [12, 11].

3 Approach

In this section we present our QoS-adaptive framework for dynamic reconfiguration of real-time systems that fulfills identified requirements R1-R6. We first review main characteristic of the ACCORD approach, as they were presented in our previous work. Then we present extensions to the approach that enable dynamic reconfiguration of a real-time system and discuss how QoS guarantees are maintained under reconfiguration.

3.1 ACCORD

ACCORD [6] prescribes that real-time systems should first be decomposed into a set of components followed by decomposition into a set of aspects. Aspects are properties of a system affecting its performance or semantics, and crosscutting the functionality of a system.

ACCORD provides a real-time component model, denoted RTCOM, to support reconfigurability [6]. RTCOM components are “grey” as they are encapsulated in interfaces, while changes to their behavior can be performed in a controlled way by aspect weaving. The process of weaving is typically done off-line and it generates the modified source code of a component as specified by aspects. An aspect consist of pointcuts, describing places in the code of components where aspect should be woven, and advices, defining code that is to be woven.

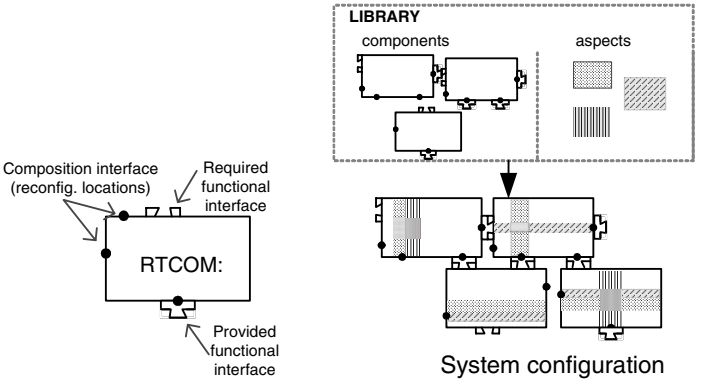


Fig. 1. (a) Reconfigurable real-time component model (RTCOM), and (b) ACCORD system configuration

Each RTCOM component has two types of functional interfaces: provided and required (see figure 1(a)). Provided interfaces reflect a set of operations that a component provides to other components, while required interfaces reflect a set of operations that the component requires from other components. Composition interfaces define reconfiguration locations in the component code. Reconfiguration locations define the points in the component code where additional modification of components can be done by aspect weaving. These points can be used by the component user (or component developer) to reconfigure a component for a specific application. Reconfiguration locations are also used for analysis purposes.

Figure 1(b) illustrates the way a system is configured and deployed in the underlying environment. Here, the composition and configuration of the system via aspects and components is done at compile time. The final system deployed in the environment is monolithic and cannot be reconfigured on-line.

3.2 Enabling Dynamic Reconfiguration

To enable dynamic reconfiguration we extend the original ACCORD approach in two ways. Firstly, we extend the RTCOM component model to ensure preservation of component and aspect states during reconfiguration. Secondly, we introduce a middleware layer, denoted ACCORD component framework, which handles communication among components and aspects.

Note that, in the context of ensuring real-time performance, an exchange of a component encompasses all the dimensions of the dynamic reconfiguration as it includes also a removal of the old version of a component and the addition of a new version. Therefore, in the following, we focus primarily on explaining the component exchange[2] part of the dynamic reconfiguration.

[2] A component exchange in the literature is also referred to as a live update of a system.

Extensions to RTCOM. To be able to preserve the internal state of components and aspects under reconfiguration, RTCOM is extended as follows. The provided interface of a component is extended with two mandatory operations, `export` and `import`. The `export` operation enables a component under reconfiguration to export its state, i.e., to store its state outside its data space (in the ACCORD component framework). The `import` operation ensures that the exported state of a component, which is to be replaced, is correctly imported into a new version of the component.

Enabling aspect exchange implies changing the way aspects are implemented within RTCOM. To conform to the dynamic aspect reconfiguration, a feature not supported by current aspect languages, and still enable general use of our framework with any of the aspect and/or component languages, we augment aspects with provided and required interfaces. These interfaces play an important role in communication and reconfiguration as we describe later in this section. Observe also that having interfaces defined for each aspect provides better encapsulation of aspect functionality and still preserves the crosscutting nature of an aspect.

System composition out of components and aspects now consists of the following steps (see figure 2(a)). First, aspects are woven off-line into components they affect. Then, using information stored in interfaces of components and aspects, these are translated into run-time entities (shared objects) recognized by the component framework. These run-time entities are then deployed onto the component framework. Conceptually, this means that weaving and run-time entity generation are done on a separate computer, and the obtained run-time entities are uploaded in the system running on another computer. Having components and aspects mapped into run-time entities that communicate via the component framework enables easy reconfiguration of both components and aspects, satisfying requirement R4.

ACCORD Component Framework. The ACCORD component framework is a middleware layer between a run-time environment and the components, which handles communication among components and aspects using jump tables (inspired by [10]). The jump table contains the list of pointers to provided and required interfaces of components and aspects. When exchanging a component or an aspect, the jump table entries for the functional interface of a component or an aspect are re-pointed to the new version of the component/aspect. As mentioned, during reconfiguration the component framework temporarily stores component and aspect internal states, using the `export` and `import` operations.

The component framework provides a user interface for reconfiguration. This reconfiguration user interface consists of operations that enable adding, removing, or exchanging components (see figure 3). During the system lifetime, reconfiguration can be requested at any time using these operations, thus, fulfilling requirement R3.

Before allowing reconfiguration to take place, the component framework always makes sure that the system can undergo reconfiguration without interrupting the task execution (line 2 in figures 3(a-c)). Tasks, which rely on operations

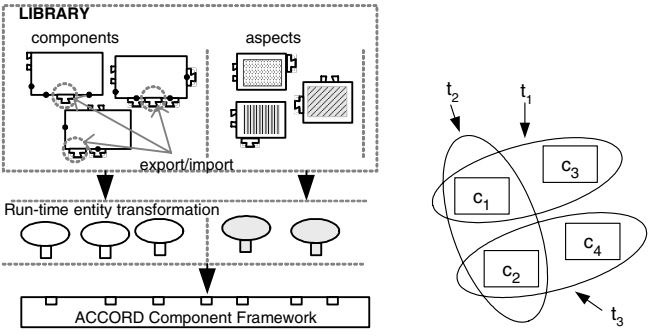


Fig. 2. (a) ACCORD system configuration revisited, and (b) an example of the relation between tasks and components

of components that are to be reconfigured, have to be completed before the reconfiguration is carried out. This is exemplified in figure 2(b), where there are four components denoted by $c_1, \ldots, c_4$, and three tasks denoted by $t_1, \ldots, t_3$. Task t_1 uses operations of component c_1 and c_3, while t_2 uses c_1 and c_2. Both t_1 and t_2 have to complete executing before c_1 can be exchanged. However, only t_1 needs to be completed before c_3 can be exchanged. Once the system can undergo reconfiguration, the actual exchange of components is initiated.

For example, the exchange of an old version of a component c, denoted c^1, with a new version, c^2, is carried out after the system is prepared for exchange, by first removing the component from the system (line 3 in 3(a)). The removal is carried out by exporting the state of (an old version) of a component into the framework (lines 3-5 in figure 3(a)). Then the jump table is re-pointed to the functional interfaces of the new version, c^2, of the component (line 4 in figure 3(b)). Finally, the new component is added to the system (line 5 in figure 3(a)). The operation `add` restores states in the new version of the component using the `import` operation (line 4 in figure 3(c)). The role of the `statePresent` variable in `remove` and `add` operations is to ensure that the state of a component is restored only if it is needed. Described reconfiguration mechanism is hidden in the component framework and it enables fast and light-weight reconfiguration, satisfying requirement R1.

```
1 exchange(c1,c2){
2   makeSystemReady();
3   remove(c1);
4   redirect(c1,c2);
5   add(c2);
6 }
```

(a) exchange

```
1 remove(c){
2   makeSystemReady();
3   if (c.hasState(){
4     state=c.export();
5     statePresent=true;
6   }
7 }
```

(b) remove

```
1 add(c){
2   makeSystemReady();
3   if (statePresent){
4     c.import(state);
5     statePresent=false;
6   }
7 }
```

(c) add

Fig. 3. Reconfiguration of a component

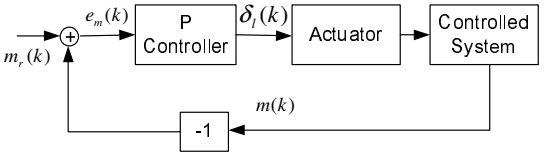


Fig. 4. Feedback control structure

The reconfiguration of any number of components can be done by a user, which can either be the system user (human) or an application (requirement R2). The component framework is aware of the version of a component, via the version number of a component, in the current configuration. When a new version is compiled into the run-time directory (the component is loaded in the memory by the framework), this is detected during application self-inspection and the component is exchanged with another version using the `exchange` operation of the component framework (see figure 3(a)). If an arbitrary number of components needs to be exchanged this can be done by invoking `exchange` operation for each of the components. This also implies that components with dependencies can be exchanged safely, in a sequence of appropriate `exchange` calls, since the tasks using these components will be completed before the exchange takes place.

When wanting to exchange one of the algorithms that crosscut the overall system, the reconfiguration can be done by exchanging aspects. Exchange is done by first weaving the desired new aspect into affected components off-line, and then employing the reconfiguration of affected components as described above[3]. Note that it is not possible to exchange aspects if they are not encapsulated into interfaces and translated into corresponding real-time entities. Although the dynamic reconfiguration of aspects is done via component exchange, the benefits of aspect-orientation are still fully retained as changes to the code that crosscut many components are still done in an automated, efficient, and modular way via aspect weaving.

3.3 Enabling QoS Adaptiveness

Recall requirements R5 and R6, stating that the system administrator must be able to specify desired system QoS and transient state system QoS in terms of worst-case QoS and how fast the QoS should converge toward the desired QoS. To fulfill R5 and R6 and, therefore, provide QoS guarantees under system reconfiguration we employ feedback control [14]. Applying feedback to control the QoS of computer systems is referred to as feedback control scheduling (FCS), and this technique has been introduced as a promising foundation for QoS control of complex real-time systems [13, 14, 15].

The desired nominal system QoS is expressed in terms of a reference QoS, which gives the level of QoS that the system must provide when it is in the steady

[3] An aspect exchange can be, from the dynamic reconfiguration perspective, regarded as exchange of a number of components with new (woven) versions.

state, i.e., when no reconfiguration is currently taking place and any effects of previous reconfiguration have passed. When a reconfiguration is taking place the system alternates to the transient state, which is characterized by fluctuations in QoS. The desired transient state QoS is usually expressed in terms of the maximum overshoot and the settling time [14]. The maximum overshoot M_p is the worst-case system QoS in a transient system state (under reconfiguration) and it is given in percentage. Hence, the maximum performance degradation of the system under reconfiguration is bounded by M_p. The settling time T_s is the time for the transient overshoot to decay and reach the steady state QoS. Hence, the settling time is a measure of system adaptability, i.e., how fast the system converges toward the desired QoS in the face of reconfiguration.

Typically, one is interested in controlling the performance of real-time systems using the metric deadline miss ratio [15], which gives the ratio of tasks that have missed their deadlines. We have employed a feedback-based QoS management method, referred to as FC-M [15], in the ACCORD dynamic reconfiguration framework. Using FC-M deadline miss ratio is controlled by modifying the admitted load. We say that a task is terminated when it has completed or missed its deadline. Let $missedTasks(k)$ be the number of tasks that have missed their deadline and $terminatedTasks(k)$ be the number of terminated admitted tasks in the time interval $[(k-1)T, kT]$. The deadline miss ratio, $m(k) = \frac{missedTasks(k)}{terminatedTasks(k)}$, denotes the ratio of tasks that have missed their deadlines.

In figure 4, the performance error, $e_m(k) = m_r(k) - m(k)$, is computed to quantize the difference between the desired deadline miss ratio, given by the reference $m_r(k)$, and the measured deadline miss ratio $m(k)$. The change to the load $\delta_l(k)$, which we denote as the manipulated variable, is derived using a P controller, hence, $\delta_l(k) = K_P e_m(k)$, where K_P is a tunable variable. The load target $l(k)$ is the integration of $\delta_l(k)$, i.e., $l(k+1) = l(k) + \delta_l(k)$. Admission control is then used to carry out the change in load, where a task is admitted into the system if the sum of the load of the task (that is waiting to be admitted) and the load of the already admitted tasks is less than the load target $l(k)$.

Consider the following example where the deadline miss ratio reference is set to 0.1 and the load threshold at the 10^{th} sampling instant is set to 0.9, i.e., $m_r = 0.1$ and $l(10) = 0.9$. A component exchange during the previous sampling interval has resulted in an increase in the execution time of the tasks, and consequently the deadline miss ratio has increased to $m(10) = 0.2$. Clearly, the component exchange has degraded the performance of the system. Therefore, we need to reduce the deadline miss ratio to the reference value $m_r = 0.1$. This is done by taking the measured value of the deadline miss ratio from the system and computing the performance error $e_m(10) = m_r(10) - m(10) = (0.1 - 0.2) = -0.1$ and then computing the change in load, i.e., $\delta_l(10) = -0.1K_P$; following the steps in the feedback structure from figure 4. The load threshold during the next sampling interval is changed to $l(11) = l(10) + \delta_l(10) = 0.9 - 0.1K_P$. The admitted load is reduced as a result of a decrease in the load threshold and, consequently, the deadline miss ratio for the 11^{th} sampling instant is reduced.

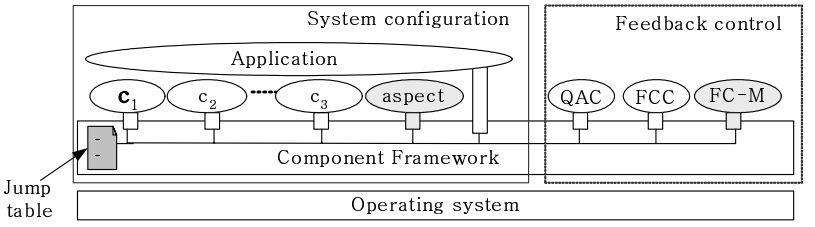


Fig. 5. ACCORD system configuration revisited

QoS management related components and aspects are implemented according to the extended RTCOM model and deployed into the ACCORD component framework. The structure of the dynamically reconfigurable system that guarantees QoS is depicted in figure 5. QoS guarantees are now satisfied in the framework by having an actuator component (AC) that implements changes in the manipulated variable and the feedback control component (FCC), which implements the control loop by measuring the controlled variable and computing the manipulated variable. The FC-M QoS algorithm is implemented as an aspect, crosscutting AC and FCC. For more details on how feedback loop can be designed using aspects and components and configured statically, as well as what types of possible QoS algorithms are supported, we refer interested readers to [16].

4 Summary

The majority of the component-based real-time systems focuses on static reconfiguration of a real-time system. However, real-time systems that are tightly coupled with their environment cannot be halted for reconfiguration to take place as their continuous operation is of paramount importance. Even if a reconfiguration takes place, affecting the temporal behavior of the system, the QoS provided by these systems must remain unaltered.

We have addressed this problem by presenting a QoS-adaptive framework for dynamic reconfiguration of real-time systems. Our framework is founded on aspectual component-based real-time system development, an approach that combines component-based and aspect-oriented software development with real-time system development. Hence, the framework supports reconfiguration in terms of components and aspects, which increases the granularity of exchangeable parts. The framework also ensures that real-time performance guarantees are maintained even when the system undergoes dynamic reconfiguration. Using the QoS-adaptive framework for dynamic reconfiguration results in efficient development and software upgrading of real-time systems, conforming to strict QoS requirements that are applied to performance-critical real-time systems.

References

1. Buttazzo, G.C.: Hard Real-Time Computing Systems. Kluwer (1997)
2. van Ommering, R.: Building product populations with software components. In: Proceedings of the 24th ACM International Conference on Software Engineering (2002) 255–265
3. Stewart, D.B., Volpe, R., Khosla, P.K.: Design of dynamically reconfigurable real-time software using port-based objects. IEEE Transactions on Software Engineering **23** (1997)
4. Sandstrm, K., Fredriksson, J., kerholm, M.: Introducing a component technology for safety critical embedded realtime systems. In: Proceedings of the International Symposium on Component-based Software Engineering, Springer-Verlag (2004)
5. Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J.M., Irwin, J.: Aspect-oriented programming. In: Proceedings of the ECOOP. Volume 1241 of Lecture Notes in Computer Science., Springer-Verlag (1997) 220–242
6. Tešanović, A., Nyström, D., Hansson, J., Norström, C.: Aspects and components in real-time system development: Towards reconfigurable and reusable software. Journal of Embedded Computing **1** (2004)
7. Cerpa, A., Estrin, D.: ASCENT: Adaptive Self-Configuring sEnsor Networks Topologies. IEEE Transactions on Mobile Computing **3** (2004) 272–285
8. Engblom, J.: Analysis of the execution time unpredictability caused by dynamic branch prediction. In: Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS). (2003) 152–159
9. Isovic, D., Lindgren, M., Crnkovic, I.: System development with real-time components. In: Proceedings of ECOOP Workshop - Pervasive Component-Based Systems (2000)
10. ITEA project: (ROBOCOP deliverable 1.5) ROBOCOP ITEA project.
11. Hissam, S., Moreno, G., Stafford, J., Wallnau, K.: Enabling predictable assembly. The Journal of Systems and Software **65** (2003) 185–198
12. Daz, M., Garrido, D., Llopis, L.M., Rus, F., Troya, J.M.: Integrating real-time analysis in a component model for embedded systems. In: Proceedings of the 30th IEEE Euromicro conference (2004)
13. Amirijoo, M., Hansson, J., Gunnarsson, S., Son, S.H.: Enhancing feedback control scheduling performance by on-line quantification and suppression of measurement disturbance. In: Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS). (2005)
14. Hellerstein, J.L., Diao, Y., Parekh, S., Tilbury, D.M.: Feedback Control of Computing Systems. Wiley-IEEE Press (2004)
15. Lu, C., Stankovic, J.A., Tao, G., Son, S.H.: Feedback control real-time scheduling: Framework, modeling and algorithms. Real-time Systems **23** (2002)
16. Tešanović, A., Amirijoo, M., Björk, M., Hansson, J.: Empowering configurable QoS management in real-time systems. In: Proceedings of the Fourth ACM SIG International Conference on Aspect-Oriented Software Development (2005)

ANTS: An Evolvable Network of Tiny Sensors*

Daeyoung Kim[1], Tomás Sánchez López[1], Seongeun Yoo[1], Jongwoo Sung[1], Jaeeon Kim[1], Youngsoo Kim[1], and Yoonmee Doh[2]

[1] Information and Communications University (ICU), 119 Munji-Dong, Yuseong-Gu, 305-714, Korea
{tomas, kimd, seeyoo, jwsung, jekim, pineland}@icu.ac.kr
[2] Electronic and Telecommunications Research Institute (ETRI), 161 Gajeong-dong, Yuseong-gu, 305-700, Korea
ydoh@etri.re.kr

Abstract. As a promising technology that enables ubiquitous computing and leads IT industries of next generation, sensor networks (SN) are foreseen to expand and populate the globe in such a way as the present Internet does. In this scenario, new challenges appear with such massive deployment, as the constant interaction of nodes and networks will transform them in dynamic entities that will need to evolve with their environment. In this paper, we discuss the future of sensor networks defining the concept of evolvability and its application. We also describe ANTS, a complete new architecture for wireless sensor networks, as our implementation to solve the challenges imposed by this evolvable future.

Keywords: Sensor Networks, Evolvability, Operating Systems, Middleware, Localization, Synchronization, Security, Network Protocols

1 Introduction

Small devices with integrated sensor capabilities are gaining credit by industries and institutions all over the planet. From the work started by some visionaries several years ago, research in the field leads old skeptics and new believers into a dream of potential applications for present and future. However, the track towards SN's future has several challenges to overcome. In this near future, when sensor nodes will be embedded in everyday's life, vehicles, food, clothes and any other kind of movable products will travel around interacting with each other and stationary networks creating and destroying communication bridges which will remain transparent to regular users. Living in this close tomorrow, we will witness the expansion of the ubiquitous computing, not only with the presence of sensor devices everywhere, but also with its global interconnection. Sensor networks will be born, will grow and will reconfigure autonomously, getting upgrades from the network, adapting to changes while learning from their environment and their fellow networks. But to get to this future, we have to think about the present and how this omnipresence of sensor devices should be managed by adaptability and autonomy capabilities. Here is where the concept of evolvability and the need for evolvable architectural design comes into play.

* This research has been partially supported by ITRC project of Korea Ministry of Information and Communication.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 142 – 151, 2005.
© IFIP International Federation for Information Processing 2005

WSN applications are numerous, such as habitat [1] and health monitoring [11], environment control and actuation [2] or even parent control [6]. Some architectures have been developed as a base for this applications [3]. However, although some efforts have been done in the development of evolvability capabilities, the results are either far away from the desirable functionalities or are not applicable to the constrained WSN devices. In this paper we introduce ANTS: An evolvable Network of Tiny Sensors. ANTS is a new architecture for the development of sensor networks which functionalities are built applying the concept of evolvability. ANTS not only develops system functionalities (Section 3) but also implements application level and middleware solutions such as context aware and UPnP (Universal Plug and Play) frameworks [10].

The rest of the paper is organized as follows: Section 0 describes generally the concept of evolvability and its characteristics. Section 0 presents our specific architecture, discussing one by one all its functionalities. Finally Section 0 depicts ANTS test-beds while Section 0 concludes the paper with a summary of the introduced concepts.

2 Evolvability in Sensor Networks

According to what has been introduced in the previous section, we could call evolvability to the capability to adapt to changes, some induced by the users according to the evolution of their needs, others provoked uncontrollably by the network nature itself and the surroundings. Evolvability support for WSN implies a series of requirements that will following be introduced. A summary also can be found in Table 1.

Indeed, the ability to adapt to changes is a key characteristic for evolvable systems (i.e. WSN), but as we will describe afterwards, a deeper view requires further considerations. Anyway, as a fundamental attribute for our analysis, we will characterize *adaptability* as the capability of a system to change or be changed to fit changed circumstances. But can a system be adaptable if it doesn't consider its interaction with other systems? For this reason we also include the definition of *interoperability*, as the ability of two or more systems or components to exchange information and to use the information that has been exchanged.

If we concentrate on the system design, we find that systems aiming to provide adaptability features must be flexible in its implementation, as monolithic approaches would require a complete redesign for any minimal change. Modular architectures, however, supply the needed looseness to address the problems of flexibility, providing well defined modules for different functionalities. We call *modularity* to the characteristic of those systems which are based in a set of interchangeable components rather than a monolithic approach. In order to guarantee a feasible modularity we also define *scalability* as the characteristic of those systems that can adapt, at any time and with the minimum cost, to a set of necessities that can be changing. Note that this definition might include configuration issues, in the sense that the highest the ease of the system to be configured to adapt to the changes, the highest its scalability. Examples of scalability and modularity applied to the SN are the support for sensor node addition or removal and the creation of modular software and hardware components.

Further considerations about the mechanisms for the addition, replacement or deletion of components should be taken. We define *upgradability* as the characteristic of a system which accepts hardware or software updates without being stopped, and *dependability* as the property of a system to continue its operation at an acceptable quality despite the occurrence of unexpected errors in that upgrading process.

Table 1. Summary of System Evolvability requirements

Requirements	Referred To
Adaptability	Ability to change with environment circumstances
Interoperability	Interaction with external systems
Modularity	Component based architecture
Scalability	Ease of adaptation to changing necessities
Upgradability	Dynamic updating processes
Dependability	Tolerant to errors during the system operation

Given the characteristics exposed previously, we can now redefine the concept of evolvability as follows: Evolvable systems are those able to adapt to its internal changes or those provoked by the interoperation with other systems. To achieve this, evolvable systems should be able to scale taking advantage of the flexibility of its modular architecture, accepting the upgrade of its components while remaining dependable under any unexpected setback in this update or failure provoked by any external cause.

3 ANTS

ANTS: an Evolvable Network of Tiny Sensors, is born with the consciousness that the future of the sensor networks is going to be deeply dynamic. Thus, ANTS builds all its architecture from scratch designing made-to-measure components that will interact with each other and the environment following the evolvability patterns exposed earlier. A block diagram of ANTS architecture can be observed on Fig. 1.

But why ANTS? Unlike other much complex systems, insect colonies are able to develop complex behaviors as a whole while the intelligence of its individual members remains very limited. While developing the concept of evolvability and adaptability to the environment, we like to think about network nodes as kind of living creatures, where they can die, become alive or communicate with other members modifying their behavior or being modified. Very limited creatures, in the end, whose knowledge of the environment is useless without sharing it with other fellows, forging a general view of the network and whose global interpretation and reaction can deal the colony to the success or the disaster.

ANTS architecture aims to solve the paradox of most present systems in which implementation is based on present static needs rather than dynamic future requirements. Following we will present the most important ANTS functionalities.

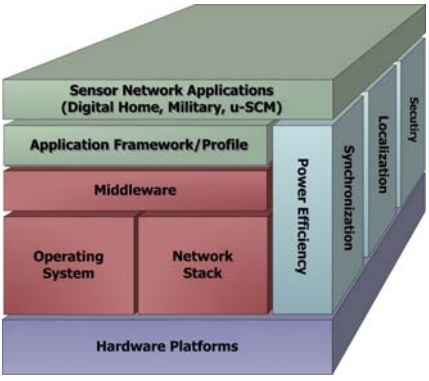


Fig. 1. ANTS architecture

Fig. 2. Prototype of ANTS H2 Outdoor. The integrated single board includes ATMEGA128L μController, CC2420 Zigbee transceiver and temperature, humidity, air pressure, light and vibration sensors.

3.1 Hardware

ANTS architecture also includes our own hardware support. The system comprises four different hardware designs (codenamed from H1 to H4) growing in complexity depending their usability needs [Section 0]. This approach provides our system with the necessary *adaptability* for different SN requirements. Thus, while the lower rank devices (H1 & H2) will include less processing capacity, small memory sizes and primitive sensors, higher level nodes (H3 & H4) will integrate 32 bit microprocessors and bigger flash memories together with more complex sensors for high level applications. Furthermore, our designs include the 802.11.5.4 standard for radio communication, used in the ZigBee specification and offering promising reductions in power consumption and memory usage. Simplest nodes aim thus to sensor readings report and routing, while the higher level nodes role is for functionalities such as base station, cluster heads and localization anchors. Fig. 2 displays a prototype of an ANTS H2 outdoor node.

3.2 ANTS Operating System

ANTS Operating System (ANTS EOS) [8] is the core of our architecture and coordinates the structural design to provide evolvability services. ANTS EOS is a multi-threaded operating system offering two kind of threads (shared-stack and self-stack) that contribute to enhance the capabilities needed for a sensor network OS. These capabilities include, amongst others, a strong concurrency model, small footprint, low power consumption, robust operation and collaborative and distributed processing. ANTS EOS also includes other advanced features , such as a flexible Hardware Abstraction Layer (i.e. to accommodate different hardware nodes), a POSIX-like API set and an efficient message handling engine valid for both local and remote communication. Fig. 3 presents the ANTS EOS architecture.

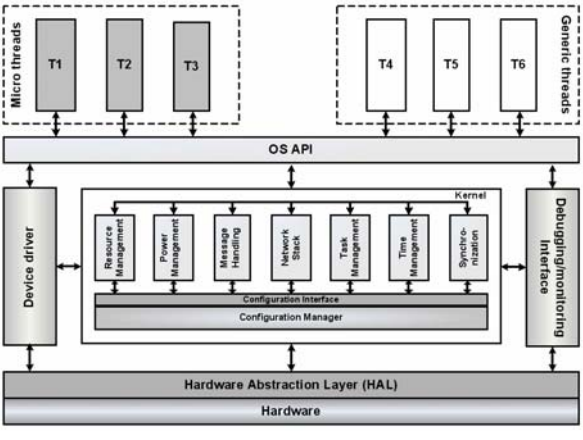


Fig. 3. ANTS EOS Architecture

The base of ANTS EOS evolvability is a modular design (implementing *modularity*) comprised of well defined modules that make use of a provided interface for upgrading processes. Capable of offering both kernel and user level upgrades, we foresee a dependable OS architecture, enforced by middleware, monitoring activities and dynamic updates able to take place remotely and without endangering the kernel or any other system functionalities (implementing *upgradability* and *dependability*).

3.3 Network Architecture

ANTS builds a two-tier, two-dimensional network architecture (NA) for optimum node communication, whose different capable nodes and different purpose network levels assure the efficiency needed for such restricted systems as sensor networks are.

On one hand, the two-tier NA is composed by two kinds of sensor nodes distinguishable by their hardware capabilities and, hence, by their potential functionalities. Together with the standard sensor network devices (micro nodes), ANTS includes more complex and powerful devices whose special resources (i.e. GPS, high bandwidth radio) make them suitable for specific functionalities such as localization anchors,

synchronization cluster heads or interoperable gateways (macro nodes). This approach offers the desired *scalability* and *interoperability* needs described previously.

On the other hand, the need for specific sensor application services builds a two-dimensional NA in which the transport domain is complemented with an additional context (application) domain for exclusive context-aware services. This way, ANTS defines the existence of an overlay network on top of the transports sensor network, which using explicit network protocols (i.e. location aimed routing protocols, [Section 0]) can offer concrete services (i.e. Context-Awareness) out of the limitations imposed by the standard transport layer (see Fig. 4 for an example on the ANTS Network Architecture).

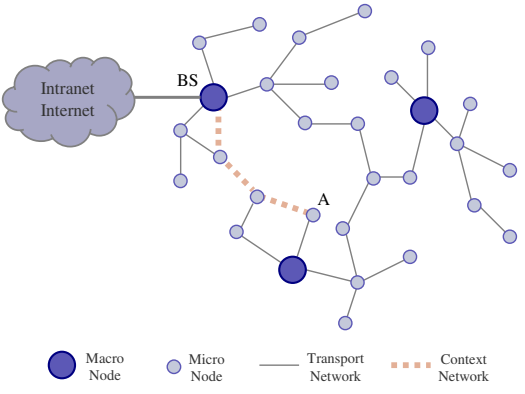


Fig. 4. ANTS Network Architecture example

Table 2. Summary of System Evolvability requirements

Protocol Name	Main characteristics
PAD [4] (Power Aware Data-centric)	-Enhanced MEPG (Minimum Energy Path Graph) -Able to provide mobility and self-healing -Hop count and power consumption as metrics -Minimizes packet delivery by being fully triggered -Supports adding or removing of nodes
PAC [9] (Power Aware Chain)	-Chain-based routing scheme -Calculates transm. costs from receiving signal strength. -Power-aware leader node election & better net. lifetime
PPVR [13] (Power-aware Position Vector Routing)	-Greedy approach to minimize flooding overhead -On-demand route discovery -Effectively avoids out-of-range communication areas -High successful rate of packet delivery.
SUMMA [7] (Sub-network Mobility support with Multi-channel Assignment)	-Temporal packet routing to mobile attached sub-networks -Gateway election & gateway data aggregation -Robust interconnection and longer life times -Multi-channel for interference and packet collision.

3.4 Communication Protocols

ANTS develops several lightweight communication protocols that efficiently deal with delay and power consumption problems applying the evolvability concept. Table **2** shows their main characteristics. Currently we have 3 routing protocols and one mobility schema while are able to provide *scalability* (with the number of nodes) and *adaptability* & *dependability* (by supporting death of nodes and mobility).

3.5 Localization

ANTS also implements a localization algorithm which is based both on measured distances and hop information obtained by the use of position aware anchors. Para-LDL-EI (Parametric Learning-based Distance Localization with Evolution Improvement) [5] supports position information search of new nodes by taking profit of past network information and causing extremely low overhead. Para-LDL-EI provides the capability for new sensor nodes to obtain automatically its position without the need to run the localization algorithm again on live nodes. It shows excellent results in up to 50% of new deployed devices and, thus, presents promising results to support efficiently multiple generation sensor networks. The support of birth and death of nodes implements *dependability, adaptability* and *scalability.*

3.6 Time Synchronization

Maintaining accurate time information in the SN does not only provide meaningful information, but also combines the system with our real physical world, allowing, for example, savings in energy or prevention of redundant information. Local clocks of inexpensive devices such as sensor network nodes are inherently inaccurate and prone to deviations of up to several seconds per week. To maintain network requirements,

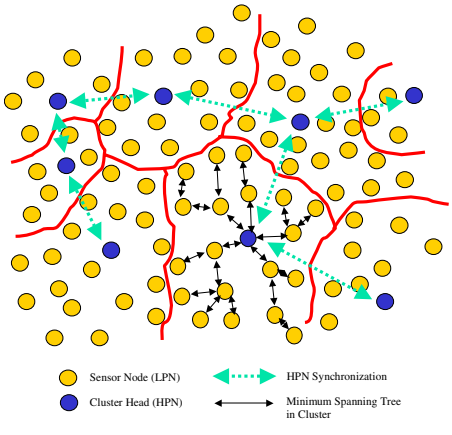


Fig. 5. Cluster-bases synchronization scheme in ANTS

periodical synchronizations of sensor nodes are required, but problems with specifying reference clocks and dealing with network time forwarding inaccuracies increasing with number of hops arise. We propose a clustering based synchronization architecture in which higher level High Precision Nodes (HPN), with accurate timing information and less frequent time synchronization needs, synchronize periodically with lower level Low Precision Nodes (LPN). Our algorithm is capable of building synchronized sensor networks from a single cluster-head reference clock, transmitting its timing information to other cluster heads and defining minimum spanning trees in cluster areas for least hop count and synchronization error. Fig. 5 shows an example of sensor field clustering, where one of the cluster is detailed with cluster-level synchronization.

3.7 Security

Existing approaches to SN security, which are mostly focused on cryptographic or multiple-path prevention schemes, have the problem that they just satisfy a part of the security requirement obviating other necessary concerns such as intrusion detection and certain key management techniques. We propose a trust evaluation based security framework for SN [12], which effective decisions on security scheme selection are based on an abstract node behavior coming from the network communication packets. Incoming packets are analyzed and their characteristics mapped to trust evaluation parameters, which will be computed locally or sent to the base station in order to build global and local trust relationships. Then the SN, considering all these trust values and relationships, will be easily able to make effective decisions in areas such as routing taking into account security concerns. This way we pretend to achieve the best combination of security level, performance, robustness, efficiency and deployment cost for SN. Fig. 6 presents a basic block diagram of the ANTS security framework.

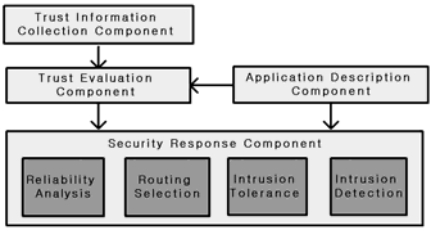


Fig. 6. ANTS Security Framework

By computing new trust values dependent on incoming packets our security framework achieves *adaptability*. Additionally, by allowing the introduction of new security values at application level, *scalability* and *modularity* are realized, while *dependability* is accomplished by supporting activity values for dead nodes.

4 ANTS Test-Beds

The ANTS team has developed two different application platforms as test-beds for our work: ANTS Smart Home and ANTS Surveillance. A graphical representation of the ANTS Surveillance test-bed is shown at Fig. 7: A helicopter deploys micro and macro sensors nodes in a target area aimed to detect the presence and movement of vehicles. The SN, aerial vehicles and other devices connect via an AODV (Ad-hoc On-demand Distance Vector) based network, forwarding the information and obtaining other services such as accurate GPS information (DGPS station).

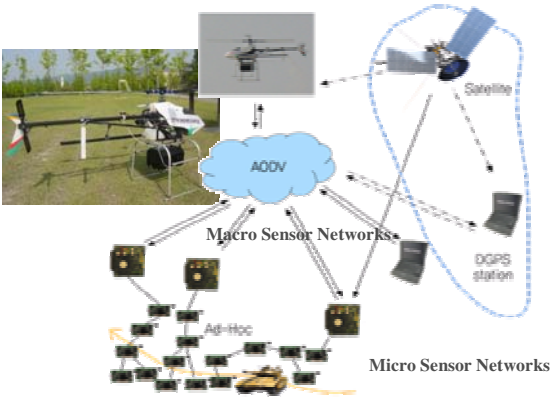


Fig. 7. ANTS Surveillance application platform

5 Conclusions

We have presented the concept of evolvability, and how the new emerging WSN technologies will need to apply it to fulfill the needs for their future prospected applications. Our particular project, ANTS, collects these ideas and builds a complete architecture distinguishable by its adaptability and flexibility, becoming the first WSN system to truly evolve with its environment and its changing needs.

Acknowledgements

We would like to thank: from ICU, *Mr. Sungjin Ahn*, *Mr. Bui Duy Bach*, *Mr. Seungki Hong, Ms. Junghyun Kim, Mr. Hyunhak Kim, Mr. Sangsoo Lee, Mr. Jihoon Park, Mr. Noseong Park, Mr. Long Minh Pham, Ms. Do Thu Thuy, Mr. Hyungjoo Song* and *Ms. Zhiying Yao*; from ITRC *Prof. Joongsoo Ma*; from Samsung *Mr. SeungRyong Park* and *Mr. K.S Chang*, and from ETRI *Dr. J.S. Jang, Dr. Jaedoo Heo* and *Mr. S. Park*.

References

1. A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler and J. Anderson: Wireless Sensor Networks for Habitat Monitoring, WSNA'02, September 28, 2002, Atlanta, Georgia, USA.
2. D. Wang, E. Arens, T. Webster and M. Shi: How The Number And Placement Of Sensors Controlling Room Air Distribution Systems Affect Energy Use And Comfort, Center for Environmental Design Research, University of California, Berkeley.
3. J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler and K. Pister: System Architecture Directions for Networked Sensors, ASPLOS IX, November 2000, Cambridge, MA, USA
4. N. Park, D. Kim, Y. Doh, S. Lee and J. Kim: Power Aware Data-centric Routing Protocol for Sensor Networks, Submitted to SCCI'05
5. D. Bach and D. Kim: A Localization Algorithm Extension for The Evolvable Sensor Network, PDCS 2004, Cambridge, MA, USA
6. M. Srivastava, R. Muntz, M. Potkonjak: Smart Kindergarten: Sensor-based Wireless Networks for Smart Developmental Problem-solving Environments, SIGMOBILE 7/01, Italy
7. J. Park, D. Kim: SUMMA: Sub-network Mobility support using Muli-channel Assignment in wireless sensor networks, Master Thesis, School Engineering, ICU 2005
8. T. Do, D. Kim, T. Sanchez, H. Kim, S. Hong, M. Pham, K. Lee, S. Park: An Evolvable Operating System for Wireless Sensor Networks, IEHSC, Singapore, May 10-13, 2005.
9. M. Pham, D. Kim, Y. Doh, S. Yoo: Power Aware Chain Routing Protocol for Data Gathering in Sensor Networks, ISSNIP 2004, , Australia, Dec. 14-17, 2004.
10. H. Song, D. Kim: Sensor Network Management System using UPnP, KISS Fall Conference, Oct. 2004.
11. Victor Shnayder et all.: Sensor Network for medical care, Technical Report, Harvard University, 2005
12. Z. Yao, D. Kim, I. Lee, K. Kim, J. Jang: A Security Framework with Trust Management for Sensor Networks, IEEE/CREATE-NET SecQoS 2005, Greece , September 9, 2005
13. S. Lee, D. Kim, S. Ahn, N. Park, " Power-aware Position Vector Routing for Wireless Sensor Networks," To appear in USN'2005, Nagasaki, Japan, 6-9 December 2005

Design Models for Reusable and Reconfigurable State Machines

Christo Angelov, Krzysztof Sierszecki, and Nicolae Marian

Mads Clausen Institute for Product Innovation, University of Southern Denmark,
Grundtvigs Alle 150, 6400 Soenderborg, Denmark
{angelov, ksi, nicolae}@mci.sdu.dk

Abstract. The widespread use of embedded systems mandates a rigorous engineering approach towards embedded software development, i.e. model-based design of embedded software. The paper presents design models of reusable and reconfigurable state machines that have been conceived in the context of the COMDES framework and in particular - the State Logic Controller and the Hybrid State Logic Controller, whose principles of operation are presented in the paper. The latter has been instrumental in developing a reconfigurable executable component, i.e. a function block of class State Machine, which can be used to implement a broad range of embedded applications such as sequential, continuous and hybrid control systems, as well as complex systems specified with hierarchical and concurrent state machines.

1 Introduction

State machines play an important role in embedded software design and recent research has focused on the efficient software implementation of state machines in terms of various types of design patterns [2, 7] and reconfigurable software components [3, 4, 9].

The conventional implementation of state machines is based on manual encoding of an abstract model such as the state transition graph, using 'switch-case' design patterns [7]. With this approach the state transition logic is built into the code, whereby a program has to be manually developed for each particular instance of the state machine. Consequently, such a program is difficult to modify and maintain. Moreover, this approach scales up badly with complex hierarchical and concurrent state machines. Therefore, a number of other design methods have been developed, e.g. The Quantum Framework [2] and the StateTable design pattern [7]. These methods provide design patterns that are potentially reusable at the source-code level but manual coding is not completely eliminated: the programmer is always required to code certain parts of the application (e.g. manual encoding of guards leading to selection of successor state and execution of related action). There are also some other problems, such as the use of sparse state transition tables resulting in considerable memory overhead, and in some cases – run-time creation of tables, which make such methods impractical for deeply embedded applications.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 152 – 163, 2005.
© IFIP International Federation for Information Processing 2005

Industrial computer systems usually adopt a different approach, whereby a software state machine is implemented by modeling the structure of its hardware counterpart, i.e. making a program that computes the state transition logic functions and executes the actions that are associated with various states. In particular, this is how sequential control programs are implemented in programmable logic controllers [9].

In both cases, conventional design methods have a major shortcoming: the resulting implementation is not reusable, because the logic of the state machine is "hardwired" in the code. Specifically, this means that a separate program has to be developed for each particular application. That might not be a problem for small state machines but it is obviously a big problem with large state machines having tens or hundreds of states. In this case the software implementation of the state machine is a time-consuming and error-prone process, and the complexity of this problem rapidly grows with the number of states and state transitions.

The above problem can be eliminated through *reusable* implementation of state machines being based on re-configurable data structures and executable components such as discrete I/O drivers, state machine drivers, function blocks, etc. [3, 4]. The resulting software construction can be viewed as a higher-level object of type 'state machine'. That object might have multiple instances depending on the contents of the encapsulated data structures (configuration tables). The latter can be configured and re-configured via a dedicated configuration tool. In this way configuration of reusable components is substituted for conventional software design, and as a result of that manual coding of state machines can be potentially eliminated.

Configuration tables may contain information representing either state machine *structure* or state machine *behaviour*. In the former case the software implementation emulates the circuit diagram of the state machine (and indirectly – the behaviour of the state machine) by computing the corresponding state variables and the associated output signals for every invocation of the control program. Industrial controllers using this technique are known as *programmable logic controllers* (*PLCs*). In the latter case the software implementation emulates the behaviour of the state machine by directly interpreting a data structure representing the state transition graph or an equivalent behavioural model, e.g. a state machine flowchart. Industrial controllers using that technique are known as *state logic controllers* (*SLCs*).

The latter approach has a number of advantages: it is easier to use as it does not require preliminary logical design of the state machine, and it is ultimately simpler to implement. Therefore, this technique has been adopted while developing a reusable and reconfigurable state machine component in the context of COMDES - a software framework for component-based design of embedded control systems [1]. The related design issues are presented in this paper, which is structured as follows: Section 2 presents design models for reconfigurable state machines, such as the State Logic Controller and its enhanced version – the Hybrid State Logic Controller. Section 3 deals with the implementation of a reconfigurable function block of class State Machine. Section 4 presents related research. A summary of the proposed software design method and its implications is given in the concluding section of the paper.

2 Design Models for Reconfigurable and Reusable State Machines

The State Logic Controller (SLC) is built around a data structure, i.e. a state transition table that contains the computer representation of the state transition graph. This data structure can be efficiently implemented as a table containing modified (multiple-output) binary decision diagrams that represent the next-state mappings Fa_1, Fa_2, … , and so on, of various states within the state transition graph – the BDD Table (see Fig. 1 and Table 1).

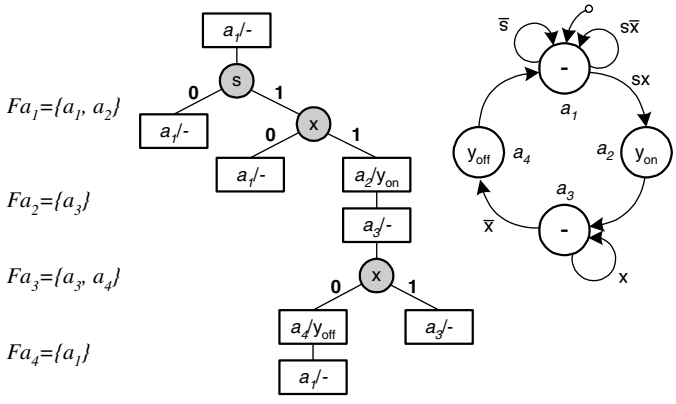


Fig. 1. Binary decision diagrams for next state mappings (Moore model)

Table 1. BDD Table of State Logic Controller for the example state machine

row	node	succTrue	succFalse	next-state mapping BDD
0	s	1	2	
1	x	3	2	Fa_1
2	a_1 / -	0	-	
3	a_2 / y_{on}	4	-	
4	a_3 / -	5	-	Fa_2
5	x	6	7	
6	a_3 / -	5	-	Fa_3
7	a_4 / y_{off}	8	-	
8	a_1 / -	0	-	Fa_4

This model is interpreted at run time, resulting in direct emulation of system behaviour: While operating, the SLC visits a number of states during successive control cycles *1, 2, 3, …* and executes the control actions associated with the visited states. These constitute a trace *P* in the state transition graph, e.g.:

$$P_i : a^i_1(1) \rightarrow a^i_2(2) \rightarrow a^i_3(3) \rightarrow \ldots , \quad (1)$$

where a^i_1 is the initial (entry) state, $a^i_2 \in Fa^i_1$, $a^i_3 \in Fa^i_2$, etc. This type of behaviour can be illustrated by an execution trace of the example state machine shown in Fig. 1:

$$a_1 \rightarrow a_1(!s) \rightarrow a_1(s!x) \rightarrow a_2(sx) \rightarrow a_3 \rightarrow a_3(x) \rightarrow a_4(!x) \rightarrow a_1 \rightarrow \ldots \quad (2)$$

Such behaviour can be implemented with a standard application-independent routine - the *state machine driver* whose algorithm is given below assuming a *synchronous*, i.e. clock-driven Moore machine with binary (on/off) inputs and outputs:

```
State machine driver is: {
  // assuming knowledge of the last state visited during
  // the previous invocation
  do {
    determine current state from among the successors of the
    previous state;
      // by processing the next-state mapping BDD of that
      // state
    execute control action associated with current state
    (if non-empty);
      // read out corresponding word of control memory and
      // generate corresponding binary (on/off) signals
  } while (not(stable_state));
}
```

The binary decision diagrams of next-state mappings Fa_1, Fa_2, Fa_3, …. are usually processed one at a time, i.e. only one BDD is processed during the current cycle, which contains the successor states of the last state visited in the previous cycle. This mode of operation results in *step-wise* execution of control actions in successive time instants *1, 2, 3,* etc., which amounts to a strictly synchronous mode of operation. In practice, the above limitation can be relaxed in a number of ways, i.e. there are specific cases that introduce asynchrony, such as wait states of unspecified duration (e.g. states a_1 and a_3 in Fig. 1) and wait states of specified duration implemented with hardware or software timers. In these two cases some states may be revisited over and over again during a sequence of control cycles, which amounts to having states whose duration is a multiple of the basic clock period.

There is yet another case that introduces a different type of asynchrony, whereby a number of states may be visited in a sequence of *immediate transitions* carried out during the current control cycle. In this case the controller executes a sequence of operations (e.g. arithmetic and comparison operations) before reaching a *stable* state, where it breaks out of the sequence and exits.

The above discussion has been made assuming binary encoding of the control memory of the state machine. However, this assumption limits the presented software design to basic sequential control applications featuring binary input and output signals and predominantly synchronous mode of operation. In a more general context it might be necessary to implement additional functionality such as timers, event counters, arithmetic and comparison operators used to compute derivative condition variables, etc.

That can be accomplished by introducing lower-level software objects called *function blocks* (FBs) implementing the above functions, whereby function block

instances may be invoked within separate states of the state machine. This model can be further generalized by making it possible to invoke not only individual function blocks, but also function block sequences specified with function block diagrams, within a given state of the state machine. Such sequences are actually implemented as *composite function blocks* (*CFBs*). CFBs may also be used to compute condition variables needed for guard evaluation within the corresponding next-state mapping BDD. This extension results in executable hybrid models - hybrid state machines (see Fig. 2). The latter may be used to specify and implement a broad range of embedded applications, such as sequential control systems with analog input signals used to compute derivative condition variables, as well as continuous and hybrid (modal) control systems.

The first type of system is illustrated with the hybrid state machine shown in Fig. 3. In fact, this is the original example discussed above, which has been re-interpreted, so as to represent a hybrid state machine for a tank pressure control system. The latter observes tank pressure and switches on a discharge valve (y_{on}) when tank pressure exceeds a predefined limit value and conversely - switches off the valve (y_{off}) when pressure is normalized. In this example the input variable s is a binary signal generated by an input driver; the condition variable x is generated by a composite function block that must be invoked whenever the choice of successor state depends on that signal, and control signals y_{on} and y_{off} are generated by two instances of FB type Binary Control (*bCtrl1* and *bCtrl2*) executed in the corresponding states – a_2 and a_4.

The conceptual algorithm of the corresponding state machine driver is given below:

```
Hybrid state machine driver is: {
  // assuming knowledge of the last state visited during
  // the previous invocation
  do {
    execute condition function block(s);
      // basic or composite FB used to compute condition
      // (guard) variables labeling transitions from the
      // previous state to successor states, if necessary
    determine current state from among successor states;
      // by processing the next-state mapping BDD of the
      // previous state
    execute control function block;
      // basic or composite FB used to compute the control
      // action associated with current state, if non-empty
  } while (not(stable_state));
}
```

The conceptual design presented in this section has been used to implement a function block of class State Machine [1]. The main feature of this type of component is its ability to invoke instances of other (basic and/or composite) function blocks inside visited states. Ultimately, it is possible for a state machine to invoke another instance of the state machine while visiting a state (OR-decomposition). In that case each instance of the state machine function block is specified by a separate BDD Table. Thus, it is possible to recursively invoke the state machine driver while processing the tables of nested state machines. Likewise, *AND*-decomposition might be implemented by invoking a sequence of such function block instances that will be executed in an interleaved fashion (i.e. one after the other) within a superstate of the upper-level state machine.

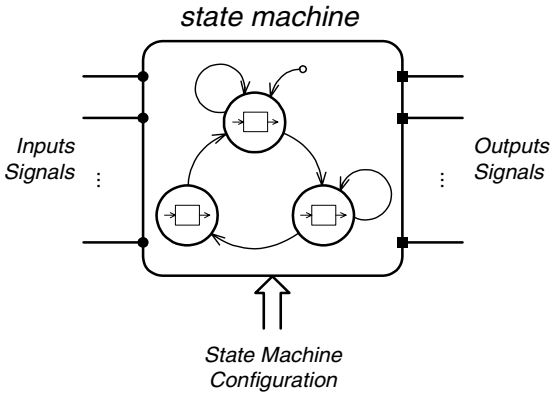


Fig. 2. Hybrid state machine

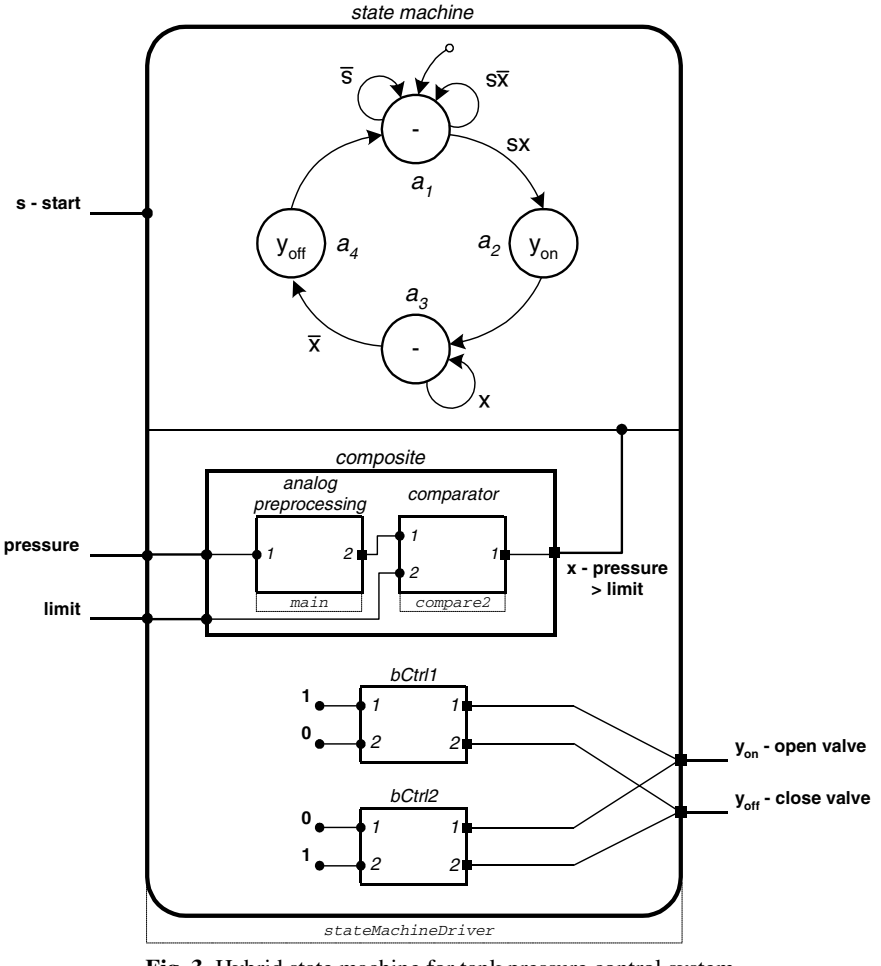


Fig. 3. Hybrid state machine for tank pressure control system

The above feature can be used to implement complex behaviours modelled by *hierarchical* and *concurrent* state machines [1]. In that case, during each invocation the program executes sequentially the constituent state machines until a *global stable state* is reached, and then breaks out until the next invocation, etc. A global stable state denotes a situation, whereby all constituent state machines have reached a stable state within the current step of execution.

Hierarchical composition of state machine function blocks provides a solution to the problems of complexity and scalability: a complex control system may be represented by a hierarchical model, whereby each constituent state machine is implemented with a different instance of that function block. An alternative approach is to decompose the object of control into functional subsystems, whereby each subsystem is assigned an individual controller modeled by a conventional (flat) state machine. In that case the overall control system may be viewed as a composition of interacting state machines that closely reflects the structure of the application [3].

3 Implementation Aspects

The algorithm given above outlines the principle of operation of the hybrid state machine driver. However, certain refinements to the algorithm need to be highlighted before discussing the actual implementation in more detail:

- The state machine driver interacts with two types of function block: function blocks that generate condition variables (*condition function blocks*) and function blocks that generate control signals (*control function blocks*). Both of them may be basic or composite function blocks (CFBs).
- The execution of condition function blocks is actually integrated with BDD processing, i.e. a FB is executed and its output signal is immediately used to make a branching decision before executing the next one, and so on, until a state node is reached,
- A condition function block might be executed only once if it generates several sequentially tested condition variables. Moreover, it is possible to test a number of such variables simultaneously via bit-patterns (mask and value) specifying the corresponding subsets of the tested variables and their values,
- The execution of condition function blocks is ultimately integrated with the execution of control function blocks, whereby BDD processing ends up with the selection of current state and the execution of a control function block associated with that state,
- It is possible to specify empty (Null) states, e.g. wait states.

It is assumed, that state transitions are complete and consistent. Completeness and consistency must be checked at configuration time, and this can be done using techniques similar to those presented in [8].

The above options have been taken into account when designing the data structures and the algorithm of the hybrid state logic controller, which are presented below. The discussion is illustrated with the example hybrid state machine given in Figs. 1 and 3.

The SLC algorithm parses a state transition table (STT), which consists of data records of the following structure:

Table 2. Description of state transition table record fields

Field Name	Field Description
type	Type of function block to be executed, index of function block type table: FBTypes[type]
function	Routine of the FB to be executed, index of function block routine: FBTypes[]→FBFunctions[function]
instance	Pointer to the FB instance execution record, argument passed to function block routine: FBTypes[]→FBFunctions[](instance)
conditionVar	Pointer to condition variable used in order to access results of function block computation
mask	AND mask imposed on condition variable
value	Value of expected masked condition variable used in order to evaluate transition guards either to True or False
successorTrue / nextState	Index pointing to successor row in case of transition guard variable evaluated to True / Index to the initial row of next state mapping in case of control action execution (state execution)
successorFalse / iTransition	Index pointing to successor row in case of transition guards evaluated to False / Flag indicating immediate transition to *nextState* in case of control action execution (state execution)

```
typedef struct {
  TFBType         type;
  TFBFunction     function;
  TFBInstance     instance;
  TConditionVar*  conditionVar;
  TConditionVar   mask;
  TConditionVar   value;
  union {
    TSTTRow       successorTrue;
    TSTTRow       nextState;
  };
  union {
    TSTTRow       successorFalse;
```

```
    TBool        immediateTransition;
  };
} TSTTRecord;
```

The records are grouped in segments representing the next-state mappings of various states of the state machine. A brief description of record fields is given in Table 2, whereby the function block is specified by three fields: *type*, *function*, and *instance*. These are used to invoke a function of a specified type on a given instance. The evaluation of transition guards to either True or False is based on the assessment of condition variables, via the *conditionVar* pointer, masked by *mask* field, and compared to required *value*. The execution of a control FB in a state is distinguished from the computation of condition variables by assigning Null to the *conditionVar* field of the STT record. Moreover, in the case of state node, *mask* and *value* fields are not used, and the meaning of the last two fields is *nextState* and *immediateTransition*. In case of transition guards evaluation all fields are used and the meaning of the last two fields is *successorTrue* and *successorFalse*.

The implementation of the hybrid state machine driver is presented below:

```
 1  row = tableRow;
 2  do {
 3    if (row->instance != NULL)
 4      FBTypes[row->type]
 5       ->FBFunctions[row->function](row->instance);
 6
 7    if (row->conditionVar != NULL) {
 8      if ( (*row->conditionVar & row->mask)
 9                                    == row->value) {
10        row = row->successorTrue;
11      }
12      else {
13        row = row->successorFalse;
14      }
15    }
16    else {
17      tableRow = row->nextState;
18      if (row->immediateTransition == FALSE) {
19        return;
20      }
21      else {
22        row = tableRow;
23      }
24    }
25  } while(TRUE);
```

STT parsing is started from the row previously saved in the *tableRow* variable (line 1). Then, the algorithm will loop until a stable state is reached (line 2-25). In the loop, first the function block routine is executed if the *instance* is specified (line 3-5). It is possible to parse the STT without executing a function block, e.g. an empty state, or a transition guard evaluation based on condition variables computed earlier (see example below). If the *conditionVar* field is not Null, evaluation of a guard variable takes place, and a successor row is chosen (line 7-15). When the *conditionVar* field is Null, the control function block associated with state has been just executed, and a state has been reached (line 16-24). If the *immediateTransition* field is equal to False a stable state has been reached and the SLC leaves the loop (line 18-20), otherwise it continues looping (line 21-23).

The presented design pattern will be illustrated with the example state machine shown in Fig. 1 and Fig. 3. Table 3 shows the encoding of the state transition table for that example, where grey rows represent condition nodes and white rows - state nodes of the BDD shown on Fig. 1.

Table 3. State transition table of example state machine

row	cond. var. / state	type	function	instance	condVar / state	mask	value	succTrue / nextState	succFalse / iTrans	next-state mapping
0	*s*	-	-	Null	&inpDriver.s	1	1	1	2	
1	*x*	TCFB	fBD	&composite	&composite.x	1	1	3	2	Fa_1
2	a_1	-	-	Null	Null	-	-	0	False	Fa_4
3	a_2	TBCtrl	main	&bCtrl1	Null	-	-	4	True	
4	a_3	-	-	Null	Null	-	-	5	False	Fa_2
5	*x*	TCFB	fBD	&composite	&composite.x	1	1	4	6	Fa_3
6	a_4	TBCtrl	main	&bCtrl2	Null	-	-	2	True	

It can be seen from the table that state nodes are differentiated by means of Null *condVar* pointers. This is because state nodes are associated with control function blocks, e.g. a_2 and a_4, or may represent empty states having a Null *instance* (e.g. a_1). Condition nodes are associated with condition function blocks, whose output location is accessed by means of a *condVar* pointer, and the corresponding output bit(s) are tested using the specified *mask* and *value* fields. A condition node may have a Null *instance* if the corresponding function block has been previously executed (e.g. the input driver FB generating signal *s*).

Rows are usually grouped in contiguous segments representing the next-state mappings (i.e. subsets of successor states) for the corresponding states of the state machine. However, the table can be eventually minimized taking into account that some of those mappings may be subsets of other next-state mappings. Here, $Fa_2 \subset Fa_3$, *and* $Fa_4 \subset Fa_1$, which is reflected in the composition of the example state transition table (see also Fig. 1).

When invoked, the state machine driver processes the table segment corresponding to the next-state mapping of the previous state (e.g. Fa_1) in order to select the current state (a_1 or a_2), execute the associated control function block and exits (if no immediate transition has been specified). During the next invocation the driver processes the segment containing the successors of the previously chosen state (e.g. Fa_2), and so on. However, in case of immediate transitions, the driver may process two or more segments in succession before a stable state is reached and the program is exited (e.g. state a_2 which is immediately followed by the stable state a_3, and likewise a_4, which is immediately followed by the stable state a_1).

4 Related Research

The increasing complexity of embedded applications has stimulated the investigation of hybrid models, e.g. mode-automata implemented in LUSTRE [5] and the hybrid models combining state machine and data flow domains in the Ptolemy II framework [6], which are similar to the hybrid state machine presented in the paper (Fig. 2). However, the above models are not implemented as reusable and reconfigurable components.

The importance of reconfigurable components, and in particular - reconfigurable state machines has been recognized, and there are already several research projects and industrial developments illustrating this approach, e.g. StateWORKS [3]. The latter employs 'virtual' finite state machines (VFSMs), which use state transition tables that are interpreted at run time by a standard routine called the VFSM executor. The system uses a non-hierarchical event-driven state machine model with a combined Moore/Mealy semantics. Consequently, an application is conceived as a hierarchy of flat state machines, which can be (re)configured by generating the necessary state transition tables. A similar approach has been developed for open machine control systems [4], where reconfigurable state machine components are once again implemented by means of state transition tables interpreted at run time by a state machine driver. This architecture uses a purely event-driven Mealy model (i.e. no guards are specified in the state transition table) and it supports hierarchical state machines in a Statecharts-like fashion.

However, these two systems do not support function blocks as defined in IEC 61131-3 and similar industrial standards [9]. Instead, the StateWORKS environment uses the concept of virtual inputs/outputs provided by a virtual I/O processor (also denoted as real-time data base). The latter consists of objects that are instances of predefined or user-supplied object classes. Likewise, the open machine control architecture employs user-supplied functions invoked from within the state machine driver in order to execute output actions. Another difference is in the data structures used to implement the state transition tables. These are relatively complex in both cases, resulting in processing and memory overhead, which can be substantially reduced by using BDD-based data structures, as suggested in this paper.

The presented function block model (Fig. 2) bears certain resemblance to function blocks defined in standard IEC 61499, which also incorporates an execution control state machine. However, that state machine is "hardwired" in the function block, i.e. it uses predefined sets of inputs and outputs and its configuration cannot be changed without reprogramming.

5 Conclusion

The paper has presented design models of reconfigurable state machines that have been conceived in the context of the COMDES framework, and in particular - the state logic controller and the extended (hybrid) state logic controller, whose principles of operation are presented in the paper. The latter has been used to develop a generic reusable component, i.e. a function block of class State Machine, which can be used

to implement a broad range of embedded applications, such as sequential, continuous and hybrid control systems.

The main idea of the proposed method is to provide a universal *executable* component that can be reconfigured *without any re-programming*. This is accomplished by updating the supporting data structure, i.e. a state transition table, whereas the executable code remains unchanged and may be stored in permanent memory. The state transition table consists of multiple-output binary decision diagrams (BDDs) that represent the next-state mappings of various states and the associated control actions. This solution has important implications:

- BDDs allow for compact encoding of state transition tables resulting in considerable memory savings in comparison with other design methods,
- BDDs allow for extremely fast processing of state transition tables via guided execution/testing of condition function blocks while evaluating a single guard specifying a transition to a successor state, thus avoiding the need to compute multiple guards in order to determine the successor state,
- This has also safety implications as long as BDD processing always results in the selection of a successor state, whereas other methods are prone to software errors that may result from incomplete and inconsistent specification of state transitions.

The presented function block design method has been validated in a number of real-time control experiments and most notably – a modal control system for a DC motor (speed and direction control) and a sequential control system for a complex plant specified in the Production Cell case study [5]. A configuration and analysis tool supporting the method is now under development.

References

1. Angelov, C., Sierszecki, K.: A Software Framework for Component-Based Embedded Applications. Proc. of the Asia-Pacific Software Engineering Conference APSEC'2004, Busan, Korea (2004)
2. Samek, M.: Practical Statecharts in C/C++: Quantum Programming for Embedded Systems. CMP Books (2002)
3. Wagner, F., Wolstenholme, P.: Modeling and Building Reliable, Re-usable Software. Proc. of the 10th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, Huntsville, USA (2003)
4. Wang, S., Shin, K. G.: Constructing Reconfigurable Software for Machine Control Systems. IEEE Trans. on Robotics and Automation, vol. 18, No 4, August (2002), 475-486
5. Maraninchi, F., Remond, Y.: Applying Formal Methods to Industrial Cases: the Language Approach (The Production-Cell and Mode-Automata). Proc. of the 5th International Workshop on Formal Methods for Industrial Critical Systems, Berlin (2000)
6. Lee, E.: Embedded Software – an Agenda for Research. UCB ERL Memorandum M99/63, University of California at Berkeley, USA (1999)
7. Douglass, B., P.: Real-Time UML: Developing Efficient Objects for Embedded Systems. Addison Wesley (1998)
8. Heimdahl, M. P. E., Leveson, N.G.: Completeness and Consistency Analysis of State-Based Requirements. IEEE Transactions on Software Engineering, TSE 22(6), (1996), 363-377
9. John, K-H., Tiegelkamp, M.: IEC61131-3: Programming Industrial Automation Systems. Springer (2001)

Optimizing Nested Loops with Iterational and Instructional Retiming*

Chun Xue[1], Zili Shao[2], MeiLin Liu[1], Mei Kang Qiu[1], and Edwin H.-M. Sha[1]

[1] University of Texas at Dallas,
Richardson, Texas 75083, USA
{cxx016000, mxl024000, mxq012100, edsha}@utdallas.edu
[2] Hong Kong Polytechnic University,
Hung Hom, Kowloon, Hong Kong
cszlshao@comp.polyu.edu.hk

Abstract. Embedded systems have strict timing and code size requirements. Retiming is one of the most important optimization techniques to improve the execution time of loops by increasing the parallelism among successive loop iterations. Traditionally, retiming has been applied at instruction level to reduce cycle period for single loops. While multi-dimensional (MD) retiming can explore the outer loop parallelism, it introduces large overheads in loop index generation and code size due to loop transformation. In this paper, we propose a novel approach, that combines iterational retiming with instructional retiming to satisfy any given timing constraint by achieving full parallelism for iterations in a partition with minimal code size. The experimental results show that combining iterational retiming and instructional retiming, we can achieve 37% code size reduction comparing to applying iteration retiming alone.

1 Introduction

Constant advance in gate capacity and reduction in chip feature size made it possible to implement system-on-chip (SoC) that allows parallel embedded systems. Parallel embedded systems, typically with VLIW or multi-core architecture, are becoming more widely used in computation-intensive applications. A large group of such applications are multi-dimensional problems, i.e., problems involving more than one dimension, such as computer vision, high-definition television, medical imaging, and remote sensing. There are usually stringent requirements in timing performance and code size for these applications. How to design parallel embedded systems that conforms timing performance requirements while minimizing code size is an interesting research topic. This paper combines iterational retiming with instructional retiming, to achieve given timing requirements for nested loops with minimal code size overhead.

There has been a lot of research on program transformations to enhance parallelism for loops. Many forms of program transformations have been proposed,

* This work is partially supported by TI University Program, NSF EIA-0103709, Texas ARP 009741-0028-2001, and NSF CCR-0309461, NSF IIS-0513669, Microsoft, USA.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 164–173, 2005.
© IFIP International Federation for Information Processing 2005

a catalog of which can be found in a survey by Bacon, Graham, and Sharp [3]. Numerous techniques have been proposed for one-dimensional loops [1] [4] [5] [8]. Renfors and Neuvo have proved that there is a lower bound of iteration period for one-dimensional loops [10]. Optimal scheduling for one-dimensional loops could reach this lower bound[5], but can not do better than this lower bound. In this paper, we show that there is no lower bound of iteration period for multi-dimensional loops. Using our proposed technique, we can achieve any given timing requirement.

A lot of works also have been done for nested loops to increase parallelism. Majority of these works are based on wavefront transformation [2][7], which achieves higher level of parallelism for nested loops by changing the execution sequence of the nested loops. This sequence of execution is commonly associated with a schedule vector s, also called an ordering vector, which affects the order in which the iterations are performed. The iterations are executed along hyperplanes defined by s. When the execution of a hyperplane reaches the boundaries of the iteration space, it advances to the next hyperplane according to the direction of s. All the iterations on the same hyperplane can be executed in parallel.

Iterational retiming first partitions the iteration space into basic partitions, and then retiming is performed at iteration level so that all the iterations in each partition can be executed in parallel. In this way, we achieve higher level of parallelism while maintaining simple loop bounds and loop indexes with minimal overhead. Two main techniques are applied in iterational retiming, namely loop partitioning and retiming. Various techniques have been proposed for loop partitioning [6] [11]. most of the loop partitioning algorithm are targeting better data locality. In our algorithm, partition is used to increase parallelism. Retiming [8] [9] has been widely applied to increase instruction level parallelism. We apply the retiming technique to iterations instead of instructions, and we show that full-parallelism for iterations in each partition can always be achieved by iterational retiming. Experimental results show that our proposed technique can always reach the given timing requirement. While iterational retiming can always satisfy the timing requirement, it is best to be combined with instructional retiming. We show that when instructional retiming is combined with iterational retiming, code size is reduced 37% in average, while achieving the same given timing performance constraint.

The remainder of this paper is organized as follows. Section 2 introduces basic concepts and definitions. The algorithm for instructional timing is presented in Section 3. The algorithm for iterational retiming is proposed in Section 4. Section 5 proposes the algorithm that combines iterational retiming and instructional retiming. Experimental results and concluding remarks are provided in Section 6 and 7, respectively.

2 Basic Concepts and Definitions

In this section, we introduce some basic concepts which will be used in the later sections. First, we introduce the model and notions that we use to analyze

nested loops. Second, loop partitioning technique is introduced. In this paper, our technique is presented with two dimensional notations. It can be easily extended to multi-dimensions.

2.1 Modeling Nested Loops

Multi-dimensional Data Flow Graph is used to model nested loops and is defined as follows. A *Multi-dimensional Data Flow Graph (MDFG)* $G = \langle V, E, d, t \rangle$ is a node-weighted and edge-weighted directed graph, where V is the set of computation nodes, $E \subseteq V * V$ is the set of dependence edges, d is a function and d(e) is the multi-dimensional delays for each edge $e \in E$ which is also known as dependence vector, and t is the computation time of each node. We use d(e) = (d.x, d.y) as a general formulation of any delay shown in a two-dimensional DFG (2DFG). An example is shown in Figure 1.

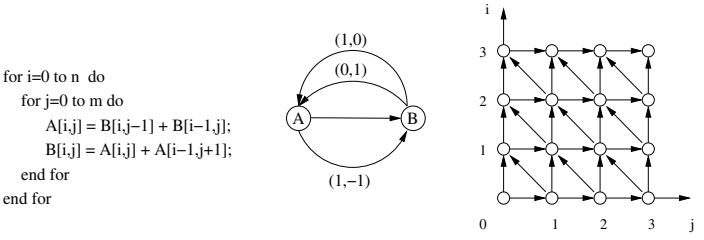


Fig. 1. A nested loop, its MDFG and its iteration space

An **iteration** is the execution of each node in V exactly once. The computation time of the longest path without delay is called the *iteration period*. For example, the iteration period of the MDFG in Figure 1 is 2 from the longest path, which is from node A to B. If a node v at iteration j, depends on a node u at iteration i, then there is an edge e from u to v, such that d(e) = j - i. An edge with delay (0,0) represents a data dependence within the same iteration. A legal MDFG must not have zero-delay cycles. Iterations are represented as integral points in a Cartesian space, called *iteration space* , where the coordinates are defined by the loop control indexes.

A **schedule vector s** is the normal vector for a set of parallel equitemporal hyperplanes that define a sequence of execution of an iteration space. By default, a given nested loop is executed in a row-wise fashion, where the schedule vector $s = (1, 0)$.

Retiming [8] can be used to optimize the cycle period of a DFG by evenly distributing the delays in it. Given a MDFG $G = \langle V, E, d, t \rangle$, retiming r of G is a function from V to integers. For a node $u \in V$, the value of $r(u)$ is the number of delays drawn from each of its incoming edges of node u and pushed to all of

its outgoing edges. Let $G_r = \langle V, E_r, d_r, t\rangle$ denote the retimed graph of G with retiming r, then $d_r(e) = d(e) + r(u) - r(v)$ for every edge $e(u \rightarrow v) \in E_r$ in G_r.

2.2 Partitioning the Iteration Space

Instead of executing the entire iteration space in the order of rows and columns, we can first partition it and then execute the partitions one by one. The two boundaries of a partition are called the *partition vectors*. We will denote them by P_x and P_y. Due to the dependencies in the MDFG, partition vectors need to be carefully chosen to ensure there is no two-way dependency so that the partition is a legal partition.

Iteration Flow Graph is used to model nested loop partitions and is defined as follows. An *Iteration Flow Graph (IFG)* $G_i = \langle V_i, E_i, d_i, t_i\rangle$ is a node-weighted and edge-weighted directed graph, where V_i is the set of iterations in a partition. The number of nodes $|V_i|$ in an IFG G_i is equal to the number of nodes in a partition. $E_i \subseteq V_i * V_i$ is the set of iteration dependence edges. d_i is a function and $d_i(e)$ is the multi-dimensional delays for each edge $e \in E_i$. t_i is the computation time for each iteration. An iteration flow graph $G_i = \langle V_i, E_i, d_i, t_i\rangle$ is *realizable* if the represented partition is legal.

3 The Instructional Retiming for Nested Loops

We propose an algorithm to do maximum instructional retiming while maintaining the original execution sequence in this section.

The instructional retiming we are applying in this setting is unique. We are retiming along one dimension of a multi-dimensional loop. This is because we wish to maintain either row-wise or column-wise execution after the retiming. The problem is equivalent to removing all the (i,j)-delays, where both i and j are none zero values, and producing a retiming such that all the edges are non-zero-delay edges if possible. If there is at least one (i,j)-delay, we can always find a retiming solution because the cycle becomes a DAG after removing (i,j)-delay edge. If there is no (i,j)-delay, then no retiming solution exists because the total delay of a cycle is a constant according to retiming property. In this case, the total delay of the cycle is always (0,k). According to retiming theory, DAG can always be retimed such that any edge has at least one delay. In the other words, a cycle with any (i, j) delay, $i > 0$, can be fully parallelized using schedule vector $\mathbf{s} = (1, 0)$. Thus, we can directly apply one-dimensional retiming to minimize the cycle period after removing the (i,j)-delay edges. Detail of our instructional retiming algorithm is shown in 3.1.

4 Iterational Retiming

In this section, we propose a new loop transformation technique, iterational retiming. First the basic concepts and the theorems related to iterational retiming are discussed. Then the procedures and algorithms to transform the loops are presented in the second section.

Algorithm 3.1 INSTRUCTIONAL-RETIME Algorithm

Require: MDFG $G = \langle V, E, \mathbf{d}, t\rangle$, schedule vector‘ $\mathbf{s}$.
Ensure: Retiming function $\mathbf{r}(v)$, retimed MDFG $G_r = \langle V, E, \mathbf{d}_r, t\rangle$, minimum cycle period $\mathbf{c}$.
/* Check the legality of $\mathbf{s}$. */
if $\mathbf{d}(e) \cdot \mathbf{s} < 0$ **then**
Report: $\mathbf{s}$ is unfeasible.
Break.
end if
/* For the feasible clock-period test algorithm */
Compute $D(u, v)$ and $T(u, v)$ for any two nodes u and v on G'.
Sort the elements in the range of $T(u, v)$.
for all Elements in the range of $T(u, v)$ **do**
Binary search to find the minimum achievable cycle period c.
/* Determine if cycle period c is feasible */
$r \leftarrow (0, 1)$; $s \leftarrow (1, 0)$;
/* Remove the edges that are not parallel to $\mathbf{r}$. */
$E' \longleftarrow E - \{e \mid \mathbf{d}(e) \not\parallel \mathbf{r}, \forall e \in E\}$
$\mathbf{d}'(e) \longleftarrow \mathbf{d}(e)/\mathbf{r}$, $\forall e \in E'$
/* The feasible clock-period test algorithm */
Compute $D(u, v)$ and $T(u, v)$ for any two nodes u and v on G'.
Construct the constraint graph according to the legality and feasible conditions of retiming with the desired cycle period c.
Use the shortest path algorithm to find a solution.
if There exists a retiming r' of G' such that $\Phi(G'_{r'}) \leq c$ **then**
/* Update the retimed graph */
$\mathbf{r}(v) \longleftarrow r'(v) \cdot \mathbf{r}(v)$, $\forall v \in V$
c feasible; Use retiming $\mathbf{r}$ to retime the MDFG G.
else
Error: Unfeasible cycle period.
end if
end for

4.1 Definitions and Theorems

Iterational retiming is carried out in the following steps. Given a MDFG, first the directions of legal partition vectors will be decided. Second, partition size will be determined to meet the input timing constraint. Third, iterational retiming will be applied to create the retimed partition.

Among all the delay vectors in a MDFG, two extreme vectors, the clockwise (CW) and the counterclockwise (CCW), are the most important vectors for deciding the directions of the legal partition vectors. Legal partition vector cannot lie between CW and CCW. In other words, they can only be outside of CW and CCW or be aligned with CW or CCW. For the basic partition in our algorithm, we choose P_x to be aligned with x-axis, and P_y to be aligned with CCW. This is a legal choice of partition vectors because the y elements of the delay vectors of the input MDFG are always positive or zero, which allows the default row-wise

execution of nested loops. For convenience, we use P_{x0} and P_{y0} to denote the base partition vectors. The actual partition vectors are then denoted by $P_x = f_x P_{x0}$ and $P_y = f_y P_{y0}$, where f_x and f_y are called partition factors, which are related to the size of the partition.

After basic partition is identified via P_x, and P_y, an IFG $G_i = \langle V_i, E_i, d_i, t_i \rangle$ can be constructed. An *iterational retiming* r is a function from V_i to Z^n that redistributes the iterations in a partition. A new IFG $G_{i,r}$ is created, such that the number of iterations included in the partition is still the same. The retiming vector r(u) of an iteration $u \in G_i$ represents the offset between the original partition containing u, and the one after iterational retiming. When all the edges $e \in E_i$ have non-zero delays, all the nodes $v \in V_i$ can be executed in parallel, which means all the iterations in a partition can be executed in parallel. We call such a partition a *retimed partition*. Properties, algorithms and supporting theorems for iterational retiming are presented below. We will first show how to choose f_x so that retimed partition can be achieved.

Given a MDFG G, let min_k be the minimum k of all the (0, k) delays in G. Let f_x be the size of partition in the x dimension. Given an Iteration Flow Graph (IFG) after we partition the iteration space with f_x and f_y, we want to make sure that the IFG can be retimed to be fully parallel using basic retiming $r = (0, 1)$. There are two types of cycles in an IFG, one with delay $d(c) = (0, y)$ and the other with delay $d(c) \geq (1, -\infty)$. The cycles with delays $d(c) \geq (1, -\infty)$ can be easily retimed to be fully parallel by using $r = (0, 1)$. But for cycles with delays $d(c) = (0, y)$, y must be $\geq$ n(c), where n(c) denotes the number of nodes in cycle c, in order to distribute (0, 1) delay to each edge in cycle c. To simplify notations, we just focus on the $(0, k)$ cycles and delays, so when we say $d(c) \geq n(c)$, it means that $d(c) = (0, y)$, $y \geq n(c)$.

Property 1. Given f_x, an edge with d(e) = (0, b) in DFG will become f_x edges in IFG from iteration node i (here we use i to denote (0, i)), $0 \leq i < f_x$, to node (i+b) mod f_x with delay = (i +b) div f_x.

Theorem 1. *If $f_x > min_k$, in the resulting Iteration Flow Graph (IFG), there exists a cycle c where $d(c) < n(c)$ and n(c) denotes the number of nodes in cycle c.*

Theorem 2. *If $f_x \leq min_k$, for each cycle c in IFG, d(c) must be $\geq n(c)$.*

As a result of the above theorems, we know that let $f_x \leq min_k$, and $r = (0, 1)$ as the retiming function, a basic partition can be retimed into retimed partition. After the iterational retiming transformation, the new program can still keep row-wise execution, which is an advantage over the loop transformation techniques that need to do wavefront execution and need to have extra instructions to calculate loop bounds and loop indexes.

4.2 The Iterational Retiming Technique

In this section, the iterational retiming algorithm is presented and explained. The complexity of the algorithm is given at the end of this section.

Algorithm 4.1 ITERATIONAL-RETIME

Require: MDFG $G = \langle V, E, d, t \rangle$, timing requirement T
Ensure: A retimed partition that meets timing requirement
/* Step 1. Based on the input MDFG, find a basic partition that is legal and have enough number of iterations to meet the timing requirement T; */
$c \leftarrow$ cycle period of MDFG ;
$P_{x0} \leftarrow (0, 1)$; /* 1.1 find P_{x0} */
$P_{y0} \leftarrow$ CCW vector of all delays; /* 1.2 find P_{y0}*/
$f_x = \{ \text{ k } | \ (0, k)$ is smallest $(0, x)$ delays $\}$; /* 1.3 find f_x*/
$f_y = \left\lceil \frac{c}{T \cdot f_x} \right\rceil$; /* 1.4 find f_y*/
$P_x = f_x \cdot P_{x0}$; /* 1.5 find P_y and P_y*/
$P_y = f_y \cdot P_{y0}$;
obtain basic partition with P_x, P_y;
/* Step 2. Call iterational retiming to transform the basic partition into a retimed partition; */
/* use r=(0,1) repeatedly to achieve full parallelism. */
Step 2.1 Apply r=(0,1) to any node that has all incoming edges with non-zero delays and at least one zero-delay outgoing edge.
Step 2.2 Since the resulting IFG is still realizable, if there are zero delay edges, go back to step 2.1.

The requirement for f_x is discussed in detail in section 4.1. We want f_x to be as large as possible. The larger f_x is, the smaller the prolog and epilog will be. Since $f_x \leq min_k$, so we pick $f_x = min_k$. Once f_x is identified, we can find f_y with the given timing requirement T and the original cycle period c. Since we need to meet the timing constraint T,

$\Rightarrow T \geq \frac{c}{f_x \cdot f_y}$
$\Rightarrow f_y \geq \frac{c}{T \cdot f_x}$
$\Rightarrow f_y = \left\lceil \frac{c}{T \cdot f_x} \right\rceil$

Let G_i be a realizable IFG, the iterational retiming algorithm transforms G_i to $G_{i,r}$, in at most $|V|$ iterations, such that $G_{i,r}$ is fully parallel.

For algorithm 4.1, in step 1, it takes $O(|V|)$ to find the cycle period, $O(|E|)$ to find P_{y0}, and $O(|E|)$ to find f_x. So it takes $O(|V| + |E|)$ to execute step 1. In step 2, it takes at most $|V|$ iterations, and each iteration takes at most $O(|E|)$ time. So it takes $O(|V||E|)$ to execute step 2. As a result, algorithm 4.1 takes $O(|V||E|)$ to complete.

5 Combining Iterational and Instructional Retiming

The COMBINE-RETIME algorithm is presented in this section. It combines instructional retiming with iterational retiming to optimize nested loops for both timing performance and code size.

Algorithm 5.1 COMBINE-RETIME

Require: MDFG $G = \langle V, E, d, t \rangle$, timing requirement T
Ensure: A retimed MDFG that meets timing requirement with smallest overheads
/* Step 1. apply instructional retiming */
if apply INSTRUCTIONAL-RETIME algorithm with $s = (0, 1)$ can satisfy T **then**
return $s = (0, 1)$;
else if apply INSTRUCTIONAL-RETIME algorithm with $s = (1, 0)$ can satisfy T **then**
return $s = (1, 0)$;
else
if $s = (1, 0)$ is legal and INSTRUCTIONAL-RETIME with $s = (1, 0)$ give a smaller cycle period **then**
apply loop index interchange;
end if
/* Step 2. apply iterational retiming reach timing requirement T */
call ITERATIONAL-RETIME algorithm;
end if

The COMBINE-RETIME algorithm first applies the INSTRUCTIONAL-RETIME algorithm in one dimension to minimize the cycle period. Row-wise execution sequence, i.e. schedule vector $s = (0, 1)$ is first attempted. Then column-wise execution sequence is attempted if schedule vector $s = (1, 0)$ is legal. When both instructional retiming failed to meet the required timing constraint T, ITERATIONAL-RETIME algorithm is called to perform iterational retiming. When column-wise execution sequence is legal and it gives a smaller cycle period after iterational retiming, loop index interchange is applied before performing iteration retiming. Applying instructional retiming reduces iteration period before iterational retiming is applied. Smaller partition size is needed in iterational retiming process, hence code size is reduced given the same timing performance requirement.

6 Experiments

In this section, we conduct experiments based on a set of DSP benchmarks with two dimensional loops: WDF (Wave Digital Filter), IIR (the Infinite Impulse Response Filter), 2D (the Two Dimensional filter), Floyd (Floyd-Steinberg algorithm), and DPCM (Differential Pulse-Code Modulation device).

Table 1 shows the code size comparison between applying iterational retiming alone and applying both iterational retiming and instructional retiming transformation, while achieving the same timing performance. In this table, code size is measure in the number of iterations included in each partition, which equals $f_x \times f_y$. Column "ITER-RE" represents the code size by applying iterational retiming alone. Column "COM-RE" represents the code size by applying both instructional retiming and iterational retiming. From this table, we can see that by applying instructional retiming before iterational retiming, we can reduce code size in an average of 37%.

Table 1. Comparison of code size between iterational retiming only and combined retiming

Bench.	**Code Size (Iterations)**					
Iteration Period = 1						
Bench.	f_x	f_y	**ITER-RE.**	f_x	f_y	**COM-RE.**
IIR	1	5	5	1	2	**2**
WDF	6	1	6	1	1	**1**
FLOYD	1	10	10	1	8	**8**
DPCM	1	5	5	1	2	**2**
2D(1)	1	9	9	1	1	**1**
2D(2)	1	4	4	1	4	**4**
MDFG1	1	7	7	1	7	**7**
MDFG2	1	10	10	1	10	**10**
Iteration Period = 1/2						
Bench.	f_x	f_y	**ITER-RE.**	f_x	f_y	**COM-RE.**
IIR	1	10	10	1	4	**4**
WDF	12	1	12	2	1	**2**
FLOYD	1	20	20	1	16	**16**
DPCM	1	10	10	1	4	**4**
2D(1)	1	18	18	1	2	**2**
2D(2)	1	8	8	1	8	**8**
MDFG1	1	14	14	1	14	**14**
MDFG2	1	20	20	1	20	**20**
Iteration Period = 1/4						
Bench.	f_x	f_y	**ITER-RE.**	f_x	f_y	**COM-RE.**
IIR	1	20	20	1	8	**8**
WDF	24	1	24	4	1	**4**
FLOYD	1	40	40	1	32	**32**
DPCM	1	20	20	1	8	**8**
2D(1)	1	36	36	1	4	**4**
2D(2)	1	16	16	1	16	**16**
MDFG1	1	28	28	1	28	**28**
MDFG2	1	40	40	1	40	**40**
Avg. Iter.			16.3			10.2
Avg Impv			**37%**			

From our experiment results, we can clearly see that combining iterational retiming with instructional retiming can reduce code size significantly while achieving the same timing performance constraints.

7 Conclusion

In this paper, we propose a new loop transformation approach that combines iterational retiming with instructional retiming to optimize both timing performance and code size. We believe iterational retiming is a promising technique

and can be applied to different fields for nested loop optimization. Combining instructional retiming and iterational retiming, we can achieve timing performance requirement for nested loops with minimal code size.

References

1. A. Aiken and A. Nicolau. Optimal loop parallelization. *ACM Conference on Programming Language Design and Implementation*, pages 308–317, 1988.
2. A. Aiken and A. Nicolau. *Fine-Grain Parallelization and the Wavefront Method.* MIT Press, 1990.
3. D. Bacon, S. Graham, and O. Sharp. Compiler transformations for high-performance computing. *Computing Srveys*, pages 345–420, 1994.
4. L.-F. Chao, A. S. LaPaugh, and E. H.-M. Sha. Rotation scheduling: A loop pipelining algorithm. *IEEE Trans. on Computer-Aided Design*, 16(3):229–239, March 1997.
5. L.-F. Chao and E.-M. Sha. Rate-optimal static scheduling for dsp data-flow programs. *IEEE Third Great lakes Symposium on VLSI*, pages 80–84, March 1993.
6. F. Chen and E.-M. Sha. Loop scheduling and partitions for hiding memory latencies. *International Symposium on System Synthesis*, 1999.
7. L. Lamport. The parallel execution of do loops. *Communications of the ACM SIG-PLAN*, 17:82–93, FEB. 1991.
8. C. E. Leiserson and J. B. Saxe. Retiming synchronous circuitry. *Algorithmica*, 6:5–35, 1991.
9. N. Passos and E. Sha. Full parallelism of uniform nested loops by multi-dimensional retiming. *Internal conference on Parallel Processing*, 2:130–133, Aug. 1994.
10. M. Renfors and Y. Neuvo. The maximum sampling rate of digital filters under hardware speed constraints. *IEEE Transactions on Cirtuits and Systems*, pages 196–202, March 1981.
11. Z. Wang, Q. Zhuge, and E.-M. Sha. Scheduling and partitioning for multiple loop nests. *International Symposium on System Synthesis*, pages 183–188, October 2001.

Realtime H.264 Encoding System Using Fast Motion Estimation and Mode Decision

Byeong-Doo Choi, Min-Cheol Hwang, Jun-Ki Cho,
Jin-Sam Kim, Jin-Hyung Kim, and Sung-Jea Ko

Department of Electronics Engineering, Korea University,
5-1 Anam-Dong, Sungbuk-ku, Seoul 136-701, Korea
sjko@dali.korea.ac.kr

Abstract. H.264 provides various useful features such as improved coding efficiency and error robustness. These features enable mobile device to adopt H.264/AVC standard to achieve effective video communications. However, the encoder complexity is greatly increased mainly due to motion estimation (ME) and mode decision. We implemented a realtime H.264 encoding system based on DSP and FPGA chip using the propose scheme to jointly optimize inter mode selection and ME using the multi-resolution analysis. In our paper, we focus on reducing the complexity while maintaining the visual quality output. Experimental results show that the proposed algorithm is over 3 times faster than the existing method while maintaining the coding efficiency and the implemented system. The implemented H.264 encoder meets the common requirement of the real-time video coding [30 frames/sec].

Keywords: H.264, inter mode decision, motion estimation, MRME, mobile video communication.

1 Introduction

Recent advances in wireless communication technology have introduced various mobile services such as multimedia message services, video on demand, and mobile video communications. Especially, there are increasing demands on mobile video communications with prevailing of the mobile devices equipped with camera module. To realize this service, video sequences have to be compressed with high coding efficiency and error robustness. The H.264/AVC is the state-of-the-art video compression standard recently developed by the ITU-T/ISO/IEC Joint Video [1].

The H.264/AVC provides various useful features such as improved coding efficiency, error robust data partitioning, and network friendliness with the network abstraction layer (NAL). These features enable mobile devices to adopt H.264/AVC standard to achieve effective video communications [2]. H.264 supports multiple reference frames and various block sizes for ME. It uses tree-structured hierarchical macroblock (MB) partitions. There are 7 different block sizes (16x16, 16x8, 8x16, 8x8, 4x8 and 4x4 blocks) that are used in a macroblock.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 174–183, 2005.
© IFIP International Federation for Information Processing 2005

The current H.264 reference software is based on a rate- distortion optimization framework for both ME and mode decision.

Among all modules in the H.264 encoder, ME and mode decision require a heavy computation of H.264 encoder, especially when rate distortion optimization is used. ME has to be performed for every MB coding mode to find the best matching block. For mode decision searches, all possible combinations of coding modes are considered to obtain the MB with the minimum cost. Moreover, since these operations are performed in multiple reference frames, the computational load significantly increases at the encoder.

The computational burden can be reduced by applying fast ME methods, such as the three step search [3], the four step search [4], the diamond search [5], and the hexagon search [6]. Recently, the UMHexagonS has been adopted for fast ME in H.264 encoder reference software (JM 84) [7]. To optimize the mode decision process, the H.264 software adopts the full mode decision algorithm (FMD) [8] inducing an exhaustive computation. For fast mode decision, the early termination technique [9] reduces the number of potential prediction modes. In [10], a classification method is proposed to reduce the average number of block types while maintaining the coding performance.

In this paper, we propose new fast ME and inter mode decision techniques using the multi-resolution analysis for the H.264/AVC encoder. For fast ME, the proposed method is based on the multi-frame/multi-resolution ME using Hexagon searching. For the fast decision of inter coding mode, we introduce a down-top merge method. The proposed method splits all MBs into 4x4 sub-MBs and merges the sub-MBs when they are classified as the same class by using a new hypothesis-and-test based method.

The organization of the paper is as follows. The proposed fast ME method is introduced in Section II. In Section III, the proposed fast inter mode decision method is given. Finally, the implemented H.264 embedded system and the simulation results are shown in Section IV and the conclusions are described in Section V.

2 The Proposed Motion Estimation Method

The multi-resolution ME (MRME) technique is an alternative ME method to the conventional block-matching algorithm. In the conventional MRME, MVs are estimated at the lowest resolution and the estimated MV is scaled appropriately to be used as an initial bias and refined for remained sub-bands on wavelet transform. However, adopting the MRME for the H.264 requires wavelet transform as well as the integer transform, which results in additional computations.

To solve this problem, we propose a modified fast integer transform that can obtain wavelet coefficients by factorizing integer transform into two steps properly. Fig. 1 shows the modified implementation of a 4x4 fast integer transform. Unlike Malvar's fast integer transform approach [11], its first step is equal to the Haar wavelet transform. The first step of the modified fast integer transform

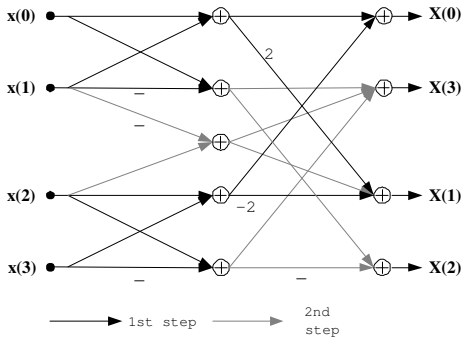


Fig. 1. Modified implementation of fast integer transform

decomposes a whole image into three layers. The coefficients in multi-layers are utilized in the proposed ME and mode decision methods.

With the modified fast integer transform, we propose a fast ME algorithm using MRME, Hexagon Searching, and Multi-frame reference. The proposed method exploits the cross correlation among multi-frames and layers of wavelet transform to reduce the computational complexity. It can achieve a smaller number of search points over other fast methods and can maintain similar or even smaller distortion error.

Fig. 2 shows the proposed searching patterns; the small square (SS), the uneven multi-hexagon-grid (UMHG), and the extended hexagon (EH). The SS and UMHG searching patterns are applied to find the coarse MV at layer 2. The EH is used to refine the MV at lower layers.

Fig. 3 shows the concept of the proposed MFMRME-HS method consisting of four steps. The detailed procedure is followed:

1. Search the MV in the object region with small motion by using the SS searching pattern at layer 2 of the reference frame (t-1) shown in Fig. 4. If the minimum sum of absolute difference (SAD) is smaller than an initial threshold, go to Step 3 to perform MV refinement at lower layers. Otherwise, go to Step 2 to keep searching the MV at layer 2. The MV obtained from Step 1 becomes the initial search center for the next step.
2. Find the MV by using the UMHG pattern at layer 2 of the reference frame (t-1) shown in Fig. 4. The MV obtained from Step 2 becomes the search center for the next step, and its corresponding minimum SAD becomes the threshold for the fast reference frame selection (FRFS) in Step 4.
3. Use the EH to refine the MV obtained from Steps 1 and 2 around the search center at lower layers. The MV refined in this step becomes a candidate of the best MV.
4. Step 2 and 3 are iterated at all the remained reference frames. For the FRFS, if the minimum SAD at layer 2 is over a threshold, MV refinement (Step 3) is not performed for its corresponding reference frame. As a result, this FRFS

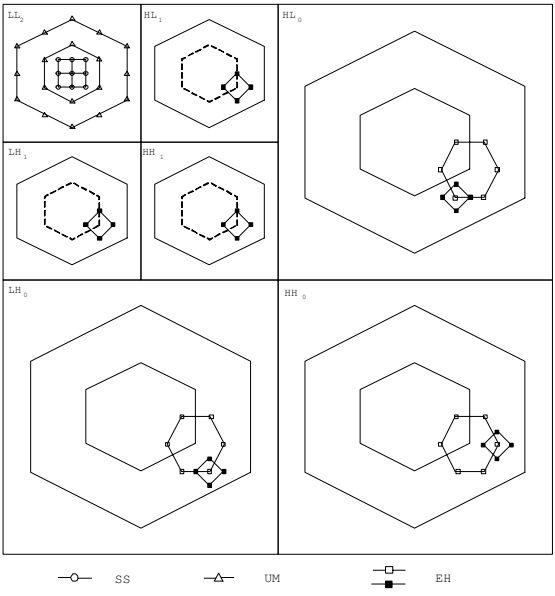


Fig. 2. Search pattern for multi-resolution

improves the performance of the proposed MFMRME-HS. Finally, select the best MV with the minimum SAD among all candidate MVs obtained from Step 3.

The proposed MFMRME-HS algorithm utilizes the correlation between multi-frames and sub-bands of wavelet transform to reduce the time complexity of ME.

3 The Proposed Fast Inter Mode Selection Method

In this section, we propose a down-top merge method for fast inter mode decision. Fig. 4 shows the proposed down-top merge method. The proposed method splits all 16x16 MBs into 4x4 sub-MBs and determines the class of each 4x4 sub-MBs by using both MVs in the LL band and the edge information. The 4x4 sub-MBs with the same class can be merged into three ways such as modes 4, 5, or 6 shown in Fig. 5, and 8x8 sub-MBs merged as mode 4 can be further grouped into one of the three block modes; modes 1, 2, and 3. After inter mode decision, ME is performed using MFMRME-HS only for the selected mode.

In order to classify 4x4 sub-MBs, we propose a new classification method based on statistical hypothesis testing. A region tends to be homogeneous if the textures in the region have very similar spatial property. Since blocks in the homogeneous region of the image can have similar MVs, they are not further split into smaller blocks. An effective way of determining the homogeneous region is

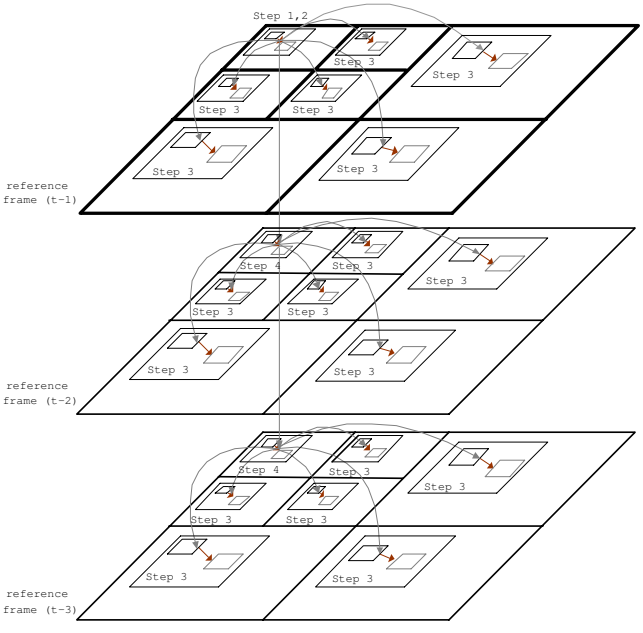


Fig. 3. Multi-frames/multi-resolution motion estimation scheme

to use the edge information, since sub-MBs in a homogeneous region have similar edge patterns.

In the proposed method, the vertical, horizontal, and diagonal edges in each sub-band of wavelet transform are used to determine the homogeneous region. The test of homogeneity hypothesis is as follows: Let $X = \{ x_0, x_1, x_2, x_3, x_4 \}$ be the feature vectors for hypothesis testing where x_0 is the absolute of motion vector in the LL band, x_1 is the average of pixel intensities in the LL band, x_2 is the sum of the amplitude of the horizontal edge in the LH band, x_3 is the sum of the amplitude of the vertical edge in the HL band, and x_4 is the sum of the amplitude of the diagonal edge in the HH band. Assume that the elements of the feature vector are independent each other. Since wavelet transform is an orthogonal transform, the amplitude of each edge is independent. Moreover, it is assumed that the distributions for the elements of the feature vector are Laplacian given by

$$f(x) = \frac{\lambda}{2} e^{-\lambda|x-\mu|} \tag{1}$$

This model is the most popular statistical model for coefficients in frequency domain and MVs [12] $\sim$ [14]. When the above assumptions are adopted, the proposed classifier can be approximated to the minimum distance classifier from

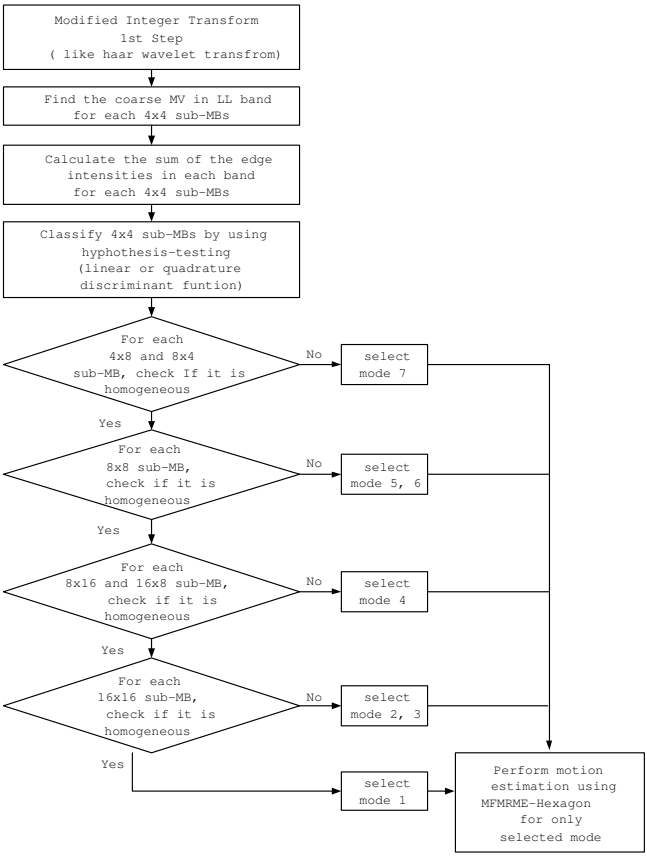


Fig. 4. Flow chart of fast inter mode decision by classifying sub-MBs

Bayes decision rule. The approximated discriminant functions for Laplacian distribution is given by

$$g_i(x) = \sum_{j=0}^{n-1} \lambda |x_j - \mu_{ij}| \tag{2}$$

where i is the index of classes and j is the index of the feature vector. We can classify each 4x4 sub-MB by calculating $g_i(x)$ for all classes and obtaining the minimum $g_i(x)$. Unlike conventional methods, the proposed inter mode decision method does not require ME and rate distortion optimization for every inter modes. However, the minimum distance classifier with a small computation is needed. Therefore, the proposed down-top merge method achieves fast inter mode decision.

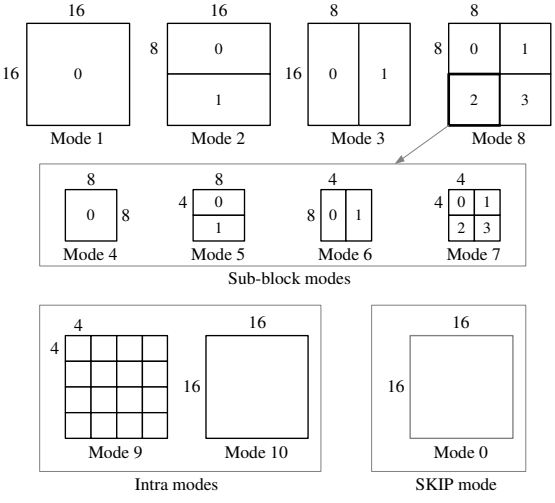


Fig. 5. Block modes of H.264/AVC

4 Experimental Results

To demonstrate the performance of the proposed system, we implemented the H.264 system which includes a 32-bit DSP processor and a FPGA, as shown in Fig. 6. The implemented H.264 encoder system targets a cost effective and high performance micro controller solution for system on chip. This system is under development with ALTERA Stratix (25,660 LEs) and TMS320C6416 (600 Mhz, 4800 MIPS, 128kb cache). Simulations have been conducted using JM 84 and the computational complexity is measured by the number of DSP clock cycles.

Fig. 7 shows the processing time of ME for multiple references. The proposed method is faster than UMHexagonS approach over 3 times since the redundant process among reference frames is reduced.

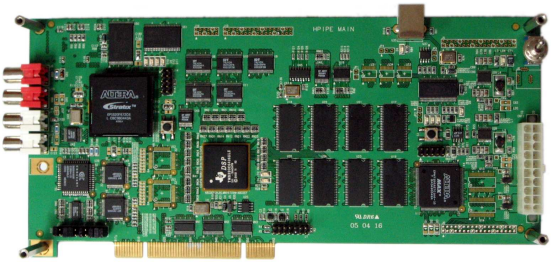


Fig. 6. The implemented H.264 system using DSP and FPGA

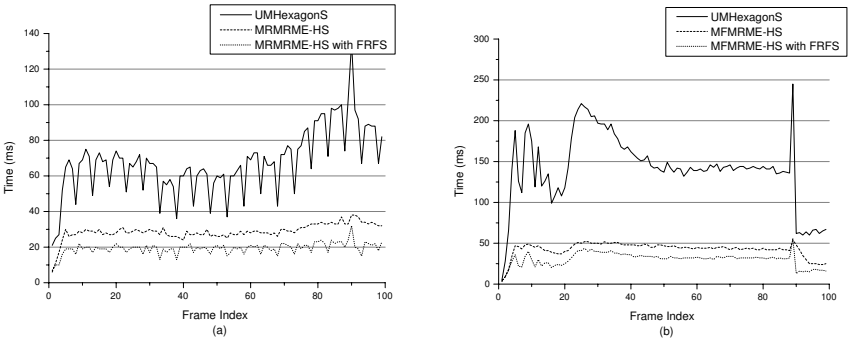


Fig. 7. Comparison of processing times. (a)"News", (b)"Table tennis".

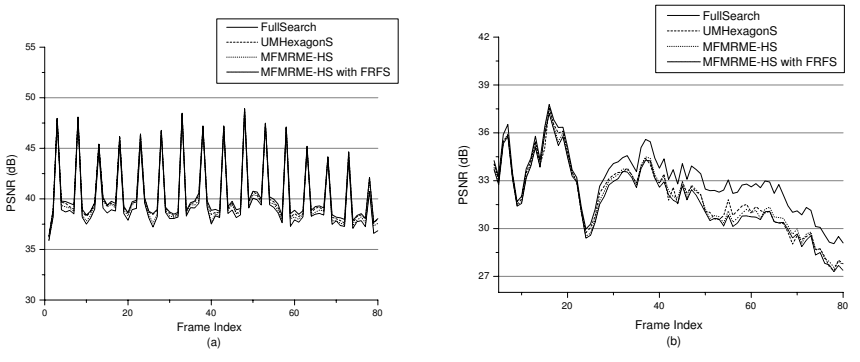


Fig. 8. Comparison of PSNR's. (a)"News", (b)"Table tennis".

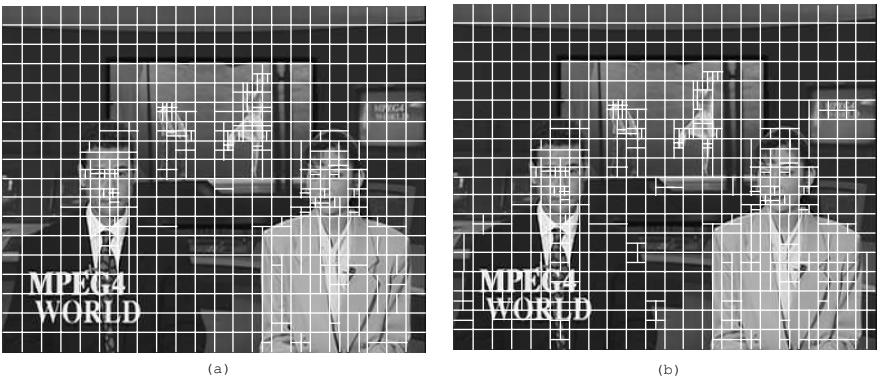


Fig. 9. Simulation results of inter mode selection. (a)FMD, (b)The proposed method.

Table 1. Test Condition Video Sequence Comparison (QP=28)

Sequence	GOP Structure	Change of PSNR(dB)	Processing Time Saving(%)
News	IPPP	-0.04	67.75
Table Tennis	IPPP	-0.02	55.25

Table 2. Test Condition Video Sequence Comparison (QP=32)

Sequence	GOP Structure	Change of PSNR(dB)	Processing Time Saving(%)
News	IPPP	-0.05	67.50
Table Tennis	IPPP	-0.03	47.75

Fig. 8 shows the change of peak-to-peak signal-to-noise ratio (PSNR) and bitrate of reconstructed images for H.264 encoder reference software with the full search, UMHexagonS search, and the proposed MFMRME-HS algorithm. The PSNR curves obtained by these methods are almost the same. As far as the processing time is concerned, the proposed method is quite efficient.

Fig. 9(a) and (b) shows simulation results of the inter mode decision with FMD and the proposed method. The modes selected by the proposed methods are close to those obtained by FMD. A group of experiments were carried out on the test sequence with 2 quantization parameters, QP = 28, 32. Table 1 and 2 show the results according to quantization parameters. The experimental results show that the proposed method reduces the encoding time by 60% on average. The PSNR loss is negligible with the highest loss at 0.05 dB.

5 Conclusions

In this paper, we have proposed a new scheme to jointly optimize inter mode selection and motion estimation using the multi-resolution analysis and show the implemented H.264 embedded system. Experimental results show the proposed MFMRME-HS method is over 3 times faster than the existing ME method while maintaining the visual quality. Moreover, the proposed mode decision method has reduced the encoding time by 60% on average. The PSNR loss is negligible with the highest loss at 0.05 dB. Experimental results indicate that the proposed fast motion estimation and inter-prediction method enable the H.264/AVC coder to be effectively adopted for the mobile video communication.

References

[1] Wiegand, T., Sullivan, G.J.: Overview of the H.264/AVC video coding standard. IEEE Trans. Circuits Syst. Video Technol. 13 (2003) 560-576 Netravali, A.N., Haskell, B.G. : Digital Pictures, 2nd ed., Plenum Press (1995) 613-651

[2] ITU-T recommendation, H.264/ISO/IEC 14496-10 AVC : Draft ITU-T recommendation and final draft international standard of joint video specification. Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVT-G050. (2003)
[3] Li, R., Zeng, B., Liou, M.L.: A new three-step search algorithm for block motion estimation. IEEE Trans. Circuits Syst. Video Technol. 4 (1994) 438-443
[4] Po, L.M., Ma, W.C.: A novel four-step search algorithm for fast block motion estimation. IEEE Trans. Circuits Syst. Video Technol. 6 (1996) 313-317
[5] Tham, J.Y., Ranganath, S., Ranganath, M., Kassim, A.A.: A novel unrestricted center-biased diamond search algorithm for block motion estimation. IEEE Trans. Circuits Syst. Video Technol. 8 (1998) 369-377
[6] Zhu, C., Lin, X., Chau, L.: Hexagon-based search pattern for fast block motion estimation. IEEE Trans. Circuit Syst. Video Technol. 12 (2002) 349-355
[7] Chen, Z., He, Y.: Fast Integer and Fractional Pel Motion estimation. Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVT-F017 (2002)
[8] Sullivan, G., Wiegand, T., Lim, K.P.: Joint model reference encoding methods and decoding concealment methods. Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVT-J049 (2003)
[9] Yin, P., Tourapis, H.C., Tourapis, A.M., Boyce, J.: Fast mode decision and motion estimation for JVT/H.264. Proc. IEEE Int. Conf. Image Processing, Barcelona, Spain. 3 (2003) 14-17
[10] Lim, K.P., Wu, S., Wu, D.J., Rahardja, S., Lin, X., Pan, F., Li, Z.G.: Fast inter mode selection. Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVT-I020 (2003)
[11] Malvar, H.S., Hallapuro, A., Karczewicz, M., Kerofsky, L.: Low-complexity transform and quantization in H.264/AVC. IEEE Trans. Circuits Syst. Video Technol. 13 (2003) 598-603
[12] Chiang, T., Zhang, Y.Q.: A new rate control scheme using quadrature distortion mode. IEEE Trans. Circuit Syst. Video Technol. 7 (1997) 246-250
[13] Jayant, N., Noll,P.: Digital coding of waveforms. Englewood Cliffs, NJ: Prentice Hall (1984)
[14] Cheung, C.H., Po, L.M.: A novel cross-diamond search algorithm for fast block motion estimation. IEEE Trans. Circuit Syst. Video Technol. 12 (2002) 1168-1177

Polyhedra-Based Approach for Incremental Validation of Real-Time Systems

David Doose and Zoubir Mammeri

IRIT - Paul Sabatier University - Toulouse, France

Abstract. Real-time embedded systems can be used in hightly important or even vital tasks (avionic and medical systems, etc.), thus having strict temporal constraints that need to be validated. Existing solutions use temporal logic, automata or scheduling techniques. However, scheduling techniques are often pessimistic and require an almost complete knowledge of the system, and formal methods can be ill-fitted to manipulate some of the concepts involved in real-time systems.

In this article, we propose a method that gives to the designer the advantages of formal methods and some simplicity in manipulating real-time systems notions. This method is able to model and validate all the classical features of real-time systems, without any pessimism, while guaranteeing the terminaison of the validation process. Moreover, its formalism enables to study systems of which we have only a partial knowledge, and thus to validate or invalidate a system still under design. This latest point is very important, since it greatly decreases the cost of design backtracks.

1 Introduction

Validation is a mandatory step in critical real-time systems design, and can be important even for non-critical ones. There are many ways to check the temporal behavior of a real-time systems: we can use formal methods based upon logic (LTL, CTL, etc.) or automata [3,2,1] (Petri nets, linear hybrid automata, etc.), or we can use scheduling techniques [7] (RMA method, etc.). Methods based on logic give precise results and have a great expressive power. They can model very specific features, but involve a strenuous designing process. Indeed, they are not initially fitted to the particular case of real-time system validation, and such notions as tasks or resources sharing do not exist in Petri nets or linear hybrid automata. Scheduling techniques have two main advantages. Firstly, they are of course completely fitted to real-time systems. Secondly, they usually have a very low complexity. However, they often are pessimistic, they require an almost complete knowledge of system parameters (tasks periods, priorities, precedences, etc.), and they give quite limited results, confined mostly to the possible schedulability of the system.

Our goal is to combine formal methods and scheduling techniques by devising a method fitted for scheduling analysis while preserving the expressivity and precision of formal methods such as linear hybrid automata. The study of linear

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 184–193, 2005.
© IFIP International Federation for Information Processing 2005

hybrid automata solvers [10,11,12] led us to the use of polyhedral domains, that enable a powerful modeling and checking. This way, we devised the only method (as far as we know) that brings together the following features:

- It can model and study classical features of real-time systems (tasks, resources sharing, precedence, atomic task executions, etc.).
- It guarantees the ending of the validation process (decidability), contrary to linear hybrid automata [13,9].
- It is non-pessimistic.
- It gives precise results on the behavior of the system: we can deduce the response times of tasks, preemptions between tasks, etc., and of course the schedulability of the system.
- It enables to validate/invalidate systems under partial knowledge. Moreover, any parameter constraint (value or linear equation on parameter value) is taken into account by the model.

The ability to check a system schedulability during its design ("under partial knowledge") can lead to important savings by avoiding backtracks. Besides, the non-pessimism implies that we can avoid system over-sizing. Lastly, since the method core works by computing all the different possible behaviors of a system, we can highlight the fact that it is similar to the use of an exhaustive set of simulations, while being far less time-consuming.

The paper is structured as follows: in section two, we introduce the notion of polyhedra and the PV-domains. In section three, we describe our method. In section four, we propose an example comparing the proposed method and a simple schedulability analysis. Finaly, we conclude in section five.

2 Notion of Polyhedra

2.1 Polyhedron

The model we propose allows to take into account any constraint based on linear inequalities. It uses polyhedra [19,14,15] to represent the parameters and the properties of the real-time system under construction.

Definition 1. *A **polyhedron** P is the intersection of a finite family of closed linear half-spaces of the form $\{x|ax \geq c\}$ where a is a non-zero row vector and c is a scalar constant.*

Double representation: A polyhedron has a dual representation: an implicit representation and a parametric representation. Indeed, a polyhedron P can be represented with a set of linear inequalities which represent its constraints. Its *implicit definition* is the following one: $P = \{x \mid Ax = b,\ Cx \geq d\}$ in which A and C are matrices and x, b and d are vectors. The *parametric representation* of the polyhedron is the following one: $P = \{x \mid L\lambda + R\mu + V\nu,\ \mu,\ \nu \geq 0,\ \sum \nu = 1\}$ In which the polyhedron is composed of several lines (columns of the matrix L), a convex combination of vertices (columns of the matrix V), and a combination

of external rays (columns of the matrix R). Algorithms exist to compute the second representation from the first one, and reciprocally. Both representations are useful. Indeed, the implicit representation is more intuitive, so we use it to represent the different constraints of the system and the results of the validation. However, the parametric representation is more efficient to compute the operations on polyhedral domains. Thus, we will use algorithms based on the dual representation, which computes both representations simultaneously.

Polyhedral domain: We mainly use two operations on polyhedra: *intersection* and *union* of polyhedra. Contrary to the intersection, the result of a polyhedra union may not be a convex polyhedron. That is why the notion of polyhedral domain is needed.

Definition 2. *A polyhedral* ***domain*** *of dimension n is defined as $\mathcal{D} : \{i | i \subset \mathcal{Z}^n,\ i \subset \mathcal{P}\} = \mathcal{Z}^n \bigcap \mathcal{P}$ where $\mathcal{P}$ is a union of polyhedra of dimension n.*

2.2 PV-Domain

We introduce a new notion that aims to create a link between the variables corresponding to the system parameters and their polyhedral representation. A **PV-domain** (Polyhedral representation of Variables) is composed of a list of variables ($\mathcal{V}$) and a polyhedral domain ($\mathcal{S}$). The variables can either be a value or a dimension of the polyhedral domain.

PV-domain operations: Adding a variable consists in adding a space dimension at the end of the polyhedral domain and adding a variable to the variables list $\mathcal{V}$ (noted: $\mathcal{D}^{'} = \mathcal{D} \oplus \{v\}$).

Removing a variable consists in removing the variable from the list and removing the corresponding dimension in the polyhedral domain $\mathcal{D}$ ($\mathcal{D}^{'} = \mathcal{D} \ominus \{v\}$).

Reducing the PV-domain consists in modifying each variable of the list representing a dimension which has a unique value in the polyhedral domain ($\forall_{v \in \mathcal{V}} \forall_{p \in \mathcal{S}}$, $\{v = a\} \subset p$) by removing the corresponding dimension in the polyhedral domain and transforming the variable representing a dimension into a variable representing the value a ($\mathcal{D}^{'} = \odot(\mathcal{D})$).

The intersection (resp. union) of the PV-domain $\mathcal{D} = (\mathcal{V}, \mathcal{S})$ and $\mathcal{D}^{'}$ is noted $\cap$ (resp. $\cup$) and defined as follows: $\mathcal{D} \cap \mathcal{D}^{'} = \left(\mathcal{V}, \mathcal{S} \cap \mathcal{S}^{'}\right)$ (resp. $\cup$).

Efficiency: The complexity and thus the computation time of intersections (resp. unions) of polyhedra depends (exponentially) on their space dimension. That is why the main objective is to reduce as much as possible the space dimension of the system model. The real efficiency of the space reduction made by the PV-domain reduction operator can be shown by simulations. The simulations on figure 1 are run on one polyhedron with ten unknown dimensions and from 20 to 500 space dimensions. In each case (i.e. for each total number of dimensions), we run 500 intersections (resp. unions) with and without using the space dimension reduction. The computation time of the unions is due to the polyhedra power-set

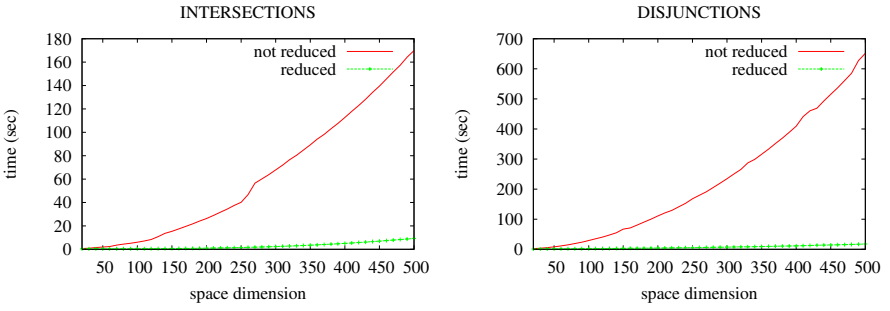


Fig. 1. PV-domain operations computation time

[5,16] operations used for the representation of the disjunctions. We can see here how efficient the dimension reduction can be, but it is important to notice that the dimension reduction is not an invertible operation.

3 The Proposed Method

System parameters: Each task instance i has four *descriptive parameters*: its period (T_i), its release time (R_i), its execution time (C_i) and its absolute deadline (D_i); and two *behavioral parameters*: the time when the task instance begins its execution (b_i), and the moment when it ends its execution (e_i). The *hidden parameters* (see below) are mandatory to represent some information specific to the system. From a geometric point of view, each descriptive, behavioral or hidden parameter corresponds to a variable of the PV-domain. Last, *static parameters* are useful to take into account some specific properties of the system. The main difference between static parameters and the others is that their values are always known.

Interactions and Behavior: In the proposed model, only two interactions are needed to represent tasks instances and their evolution. Indeed, an instance can either **preempt** or **delay** another instance. The behavior of tasks instances is defined by the interactions which can either exist between the instances. In the paper we will compute *possible behaviors*. A behavior is possible for a group of instances (see below) if all the interactions characterizing this group can occur on the instances of the group, with the respect of all the constraints of the system.

Group: A group of tasks instances is characterized by:

- a list of instances with interactions (noted *interactions*);
- the first instance in interaction of the group (*first*);
- a list of context instances (noted *contexts*);

- a PV-domain representing the properties of the previous groups (i.e. the constraints of the systems before the creation of the group noted *pre_constraints*);
- a PV-domain representing the properties of the groups (*part_system*).

Each instance of the *interactions list* is at least in interaction with another instance of the list. The first instance in interaction and the number of instances of each task are useful to determine the beginning and the end of the group. The list of context instances is made of instances which, under the same constraints, are not in interaction with any instance of the list of the instances in interaction.

3.1 Method Overview

Analysis of real-time systems with the proposed method has two main parts. The first part consists in taking into account the knowlegde of the system by: adding the tasks of the system, modifying the static parameters, adding the tasks constraints and adding the invariants. The second part consists in analysing the behaviors of the system by determining all the groups of interactions and analysing the schedulability of each group. At the end of the validation process the method provides the list of the situations in which the system is schedulable, and the list of the situations in which the system is not schedulable.

3.2 Static Steps

The first step consists in simply adding the tasks of the system in order to determine the number of tasks.

In this first step no information about the tasks are taken into account. The second step consists in setting the static parameters of the system. Thus, we determine whether the tasks are periodic, atomic, ... We also determine the scheduler category (EDF or fixed priority). At the end of this step, the static parameters of the system and the number of tasks are known, thus we can create the first polyhedral representation of the system.

In the third static step, the designer takes into account the system information by adding constraints to the domain representing the system as previously described. For example, if the worst execution time of task τ_1 is $20\,ms$, the system is equal to $system \cap (C_1 \leq 20)$. At the end of this step, the domain representing the system contains all the information about the tasks.

In the last static step, the invariants of the system concerning the tasks are taken into account by adding the corresponding constraints. If the previously added information of the system does not respect these invariants, then the domain is empty, and the validation ends. The domain representing the system is: $system = \odot(information \cap (\forall_{\tau_i}(R_i \geq 0) \cap (C_i > 0t) \cap (D_i \geq 0)))$

3.3 Determining Groups

The principle is to determine all the possible groups of instances as shown by the following algorithm:

```
groups: stack of Group ;
groups.push(Group(system)) ;
while groups.not_empty() do
    Group g = groupes.pop() ;
    if g.complete() then
        if (not g.cycle) and g.validation then
            groups.push(g.next()) ;
    else
        for each instance τ_i do
            if g.can_interact_with(τ_i) then
                goups.push((new Group()).add_interaction(τ_i)) ;
            if g.can_be_context_of(τ_i) then
                goups.push((new Group()).add_context(τ_i)) ;
```

The function *can_add_interaction* (resp. *can_be_context*) determines whether or not the instance τ_i can interact (resp. can be a context instance) in the group g. This is determined by verifying the non-emptyness of the following PV-domain representing the addition of the instance: $part_system \cap (r_{i,j} \geq r_{first}) \cap (r_{i,j} < r_{first} + c_{tot})$ (resp. $part_system \cap (r_{i,j} \geq r_{first} + c_{tot})$) while r_{first} represents the release time of the first instance of the group (i.e. the beginning of the group), $r_{i,j}$ the release time of the instance j of the task i and c_{tot} represents the length of the group.

The function *cycle* aims to determine if the the group g is in a behavioral cycle with another group that has already been computed. The notion of cycle is important because it guarantees the end of the validation process. A group G is in behavioral cycle with the group $G^{'}$ if the instance in interaction can correspond (verifyed with the function *mapping*) and if the constraints of the system are the same but at another moment. The function *validation* determines if the group of instances is schedulable or not. This function is described in the following subsection.

3.4 Group Schedulability

In this subsection, we detail the validation of a group. The tasks instances are not linked to the corresponding tasks, thus the instances parameters are noted with lowercase letters with a single index representing the instance number.

General information and invariants: The invariants are mandatory to maintain the correctness of the model. They are added like any information about the system. The following invariants are useful in the proposed method: $\forall \tau_i$, $(b_i \geq r_i) \cap (e_i \geq b_i + c_i)$

Scheduler information: In practice, two schedulers are used: fixed priority and EDF. In this paper, a lower priority value means a greater priority.

Determining priorities: In order to represent a fixed priority scheduler, the priority of a task instance just has to be equal to the priority of the corresponding task. If the priority of one or more tasks is not known, it is important to add the following information to the system: $\forall_{i,j,k} P_{i,j} = P_{i,k}$; because without this equation, two task instances of the same task can have different priorities. If the scheduler of the system is EDF, then the priority of each task instance is equal to the absolute deadline: $\forall_i p_i = d_i$.

Determinism: A scheduler is said to be deterministic if for every situation for every system with the same parameters, the same task instance is executed. The previously defined properties and hypotheses of the system and the method do not induce that the scheduler is deterministic; because two instances can have the same priority. However, the following property must be guaranteed:

Property 1. If several task instances that need the processor at the same time have the same priority which is the smallest of the system, then the scheduler must choose one and only one instance to execute.

Property 2. Once this choice done, the scheduler must not change it until the next system's change.

These two properties aim to prevent two task instances from being executed at the same time. To represent a possible non-deterministic scheduler, we introduce another parameter, the hidden parameter Π_i. This parameter acts as a hidden secondary priority, known by the scheduler only. Consequently, a task instance (i) has a higher priority than another (j) if and only if: $prio(i,j) = (p_i < p_j) \cup ((p_i = p_j) \cap (\Pi_i < \Pi_j))$. In order to verify the previous properties, the following constraint is added to the system: $\forall_{i,j}\, such\, as\, i \neq j,\ (p_i = p_j) \cap (\Pi_i \neq \Pi_j)$. The non-determinism of the scheduler comes from the fact that the hidden parameter Π_i is not given a fixed value.

Task instances behaviors: The notion of time is introduced with parameters which represent a moment when a particular action happens and thus a behavior. The complete behavior of a task instance is described with the following equation: $instance\, behavior_i = (\bigcup (possible\, b_i)) \cap (\bigcup (possible\, e_i))$.

The execution beginning of a task instance (b_j) can be *delayed* by another task instance (τ_i) with a higher priority: $delay(i,j) = prio(i,j) \cap (b_i \leq r_j) \cap (e_i > r_j)$. It is important to notice that only the task instance with the latest end of execution really delays the task instance. If the task instance i delays the task instance j, then the task instance j begins its execution when the task instance i ends: $b_j = e_i$.

The execution end of a task instance depends on the moment the task instance begins its execution and on the interactions with other task instances during its execution. In this case the interactions are *preemptions*. The end of an instance execution: $e_j = b_j + c_j + \sum_{i\, preempt\, j} c_i$. The task instance i preempts the task instance j if: $preempt(i,j) = prio(i,j) \cap (b_i > b_j) \cap (b_i < e_j)$.

Verifying properties: The result of the behavioral step represent all the possible behaviors of the system. The question consists in asking what is possible in the system behaviors by adding the correspond constraint. Thus, the PV-domain $\mathcal{D}$ representing the system is schedulable if: $\forall_{\tau_i} \mathcal{D} \cap (e_i > d_i) = \emptyset$

Specific characteristics

Atomic task instances: A task instance that cannot be preempted is said to be atomic. Thus, we introduced the static parameter $atomic(i)$ (and $not\, atomic(i)$) to indicate whether a task instance can be preempted. This induces a modification of the previously defined interactions by adding $\cup$ (resp. $\cap$) $atomic(i)$ to the equation $delay(i,j)$ (resp. $preempt(i,j)$).

Resource sharing: Sharing resources [18,6] is a complex problem in real-time systems and several algorithms exist to solve it, thus we won't study it in detail. But we want to show how easy it is for this method to handle this problem by considering mutexes. The static parameter $Res(i,r)$ indicates that the task instance i needs the shared resource r. The function $Resource(i,j)$ represents that the task instance i and the task instance j need a common shared resource. This function is defined as follows: $Resouce(i,j) = \exists r,\, Res(i,r)\, and\, Res(j,r)$. Thus, the equations of the interactions are modified:

$$delay(i,j) = (prio(i,j) \cup atomic(i) \cup Resource(i,j)) \cap (b_i \leq r_j) \cap (e_i > r_j)$$
$$preempt(i,j) = prio(i,j) \cap \neg atomic(j) \cap \neg Resource(i,j) \cap (b_i > b_j) \cap (b_i < e_j)$$

4 Example

In this section we present a simple example in which the system is made of three aperiodic tasks. The priorities of the tasks are known: $P_0 = 1$, $P_1 = 2$ an $P_2 = 3$. The deadlines are also known: $D_0 = 6$, $D_1 = 6$ et $D_2 = 20$. The release time are partially known: $R_0 = 0$, $R_1 = 3$ and R_2 is unknown. The execution times of τ_0 and τ_2 are partially known: $2 \leq C_0 \leq 5$ and $C_2 = 2$. The execution time of τ_1 depends on the execution of the other tasks. Indeed, $C_1 = 3$ if τ_1 begins its execution before the end of τ_0; $C_1 = 4$ if τ_1 begins its execution before the end of τ_2 and after τ_0; and $C_1 = 5$ if τ_1 begins its execution after τ_0 and τ_2.

To validate this system with a scheduling technique we proceed as follows: first, we determines the worst case values of the system parameters. Thus $C_0 = 5$, $C_1 = 5$ and $C_2 = 2$. This approximation induces the pessimism of this method. That is why we show that the system is not schedulable. Indeed, a simple execution trace highlights that the second task misses its deadline (figure 2).

The main advantage of our method is it can represents the specific constraint of the execution of the second task. Thus, the following equation represent this information: $((C_1 = 3) \cap (b_1 < e_0)) \cup ((C_1 = 4) \cap (b_1 \geq e_0) \cap (b_1 < e_2)) \cup ((C_1 = 5) \cap (b_1 \geq e_0) \cap (b_1 \geq e_2))$. Then, the behavioral step is done. In the result representing the possible behaviors of the system we can highlight the invariants

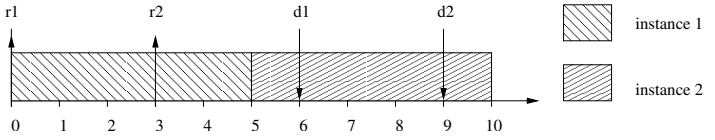


Fig. 2. Example

of the system behavior: $b_0 = 0$, $C_1 = 4$, $e_0 = C_0$. It shows that the first task is neither delayed nor preempted and also that the execution time of the second task is $C_1 = 4$. That means the second task begins its execution after the first one and before the third one, in all the possible situations. Thus, the worst value for this parameter ($C_1 = 5$) determined with the scheduling technique is impossible. The proposed method is not pessimistic; that is why we can prove that the system is schedulable with this method but not with a schedulability analysis.

With this simple example we show that it is possible to validate a real time system in which some parameters are exactly known, some are partially known, some are unknown and some parameters are defined with specific constraints. We also highlight that the proposed method is not pessimistic, contrary to a simple scheduling analysis.

5 Conclusions

We propose a new non-pessimistic method to analyse real-time systems. This method can be used to represent systems with specific behaviors and complex relationships between their parameters, because any information that can be represented with a linear inequality can be taken into account. A major advantage of this method is that can be used to analyse study a system at any design step. Indeed, this method can be used with partial knowledge and thus it is well adapted to reduce costs. To complete this work, we intend to focus on two points: taking into account new system characteristics, and implementing a complete software. Indeed, several system behaviors have to be represented in our method (dependable and/or non-preemptive tasks, resources sharing, partitioning [8,20,17,4]), which implies some modifications of the previously defined equations. We also plan to include specific components such as bounded buffers.

References

1. R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.
2. Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei-Hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, pages 209–229, 1992.

3. Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
4. B. Andersson and J. Jonsson. Fixed-priority preemptive multiprocessor scheduling: To partition or not to partition. In *Proceedings of the Int'l Conf. on Real-Time Computing and Applications*, pages 337–346, Cheju Island, Korea, December 2000. IEEE Computer Society Press.
5. R. Bagnara. A hierarchy of constraint systems for data-flow analysis of constraint logic-based languages. In *Science of Computer Programming*, pages 30(1–2):119–155, 1998.
6. T. P. Baker. A stack-based resource allocation policy for realtime. In *Real-Time Systems Symposium*, pages 191–200. IEEE Computer Society Press, 1990.
7. L. P. Briand and D. M. Roy. *Meeting Deadlines in Hard Real-Time Systems: The Rate Monotonic Approach*. IEEE Computer Society, 1999.
8. Airlines Electronic Engineering Committee. Arinc specification 653, January 1997.
9. N. Halbwachs, Y-E. Proy, and P. Raymond. Verification of linear hybrid systems by means of convex approximations. In *Static Analysis Symposium*, pages 223–237, 1994.
10. T. A. Henzinger, P-H. Ho, and H. Wong-Toi. Hytech: The next generation. In *IEEE Real-Time Systems Symposium*, pages 56–65, 1995.
11. T. A. Henzinger, P-H. Ho, and H. Wong-Toi. A user guide to hytech. In *Tools and Algorithms for Construction and Analysis of Systems*, pages 41–71, 1995.
12. T. A. Henzinger, P-H. Ho, and H. Wong-Toi. HYTECH: A model checker for hybrid systems. *International Journal on Software Tools for Technology Transfer*, 1(1–2):110–122, 1997.
13. Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? In *Proceedings of the 27th Annual Symposium on Theory of Computing*, pages 373–382, 1995.
14. Doran K.Wilde. A library for doing polyhedral operations. Technical Report 2157, December 1993.
15. Sanjay Rajopadhye Patrice Quinton and Tanguy Risset. On manipulating z-polyhedra. Technical Report 1016, Jully 1996.
16. P. M. Hill R. Bagnara and E. Zaffanella. Widening operators for powerset domains. In *Quaderno 349, Dipartimento di Matematica, Universit di Parma, Italy*, 2004.
17. J. Rushby. Partitioning in avionics architectures: Requirements, mechanisms, and assurance. Technical report, Menlo Park USA, March 1999.
18. L. Sha, R. Rajkumar, and J.P. Lehoczky. Priority inheritance protocols: An approach to real-time synchronization. *IEEE Transactions on Computers*, 39:1175–1185, September 1990.
19. H. Le Verge. A note on cherniakova's algorithm. Technical Report RR-1662, April 1992.
20. B. L. Di Vito. A formal model of partitionning for integrated modular avionics. Technical report, August 1998.

Checkpointing for the Reliability of Real-Time Systems with On-Line Fault Detection*

Sang-Moon Ryu and Dong-Jo Park

Korea Advanced Institute of Science and Technology,
373-1 Guseong-dong, Yuseong-gu, Daejeon 305-701, Republic of Korea
smryu@kaist.ac.kr, djpark@ee.kaist.ac.kr

Abstract. The checkpointing problem in real-time systems equipped with on-line fault detection mechanisms is dealt with from a reliability point of view. The reliability analysis is performed with the assumption that transient faults occur in accordance with a Poisson process and are detected immediately by the detection mechanisms. And the best equidistant checkpointing strategy that maximizes the reliability of the system against transient faults is derived.

1 Introduction

Transient faults in semiconductor devices are becoming more significant because of increased density, low supply voltage, fast switching signals and so on [1]. Checkpointing is a well known technique to overcome transient faults in computer systems. It means periodically saving the state of a task in a safe storage place. When the manifestation of a transient fault is detected, the state of the affected task will be restored to the state stored at the latest checkpoint. This process is called rollback-recovery. The specific points at which checkpointing is performed are called checkpoints and the length of the time between two successive checkpoints is said to be a checkpoint interval.

Many papers dealt with the problems of checkpointing in real-time systems from various points of view [2,3,4,5,6]. In this paper, the reliability problem of equidistant checkpointing, which relies on the use of a constant checkpoint interval, in a single task real-time system under transient faults is explored. The transient faults are assumed to occur according to a Poisson process and be detected by on-line detection mechanisms [7,14,15,16] with no latency. The reliability of the system with equidistant checkpointing is analyzed and the best checkpointing strategy is derived, which achieves the maximum probability of successful task completion with given parameters such as task execution time, available slack time, checkpointing and recovery overheads.

The following assumptions were made in this work, which were used in other literature:

- Transient faults occur according to a Poisson process with rate λ, which is common in many papers dealing with transient faults [2,3,4,5,8,10,11,12,13].

* This work was supported by SaTReC of KAIST.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 194–202, 2005.
© IFIP International Federation for Information Processing 2005

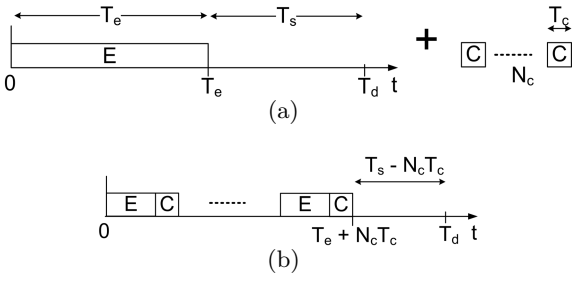


Fig. 1. The task (a) before inserting checkpoints (b) after inserting checkpoints

- The effect of a transient fault disappears during the corresponding recovery operation.
- The manifestation of transient faults is perfectly detected by the detection mechanisms [3,4,5,10]. If necessary, the fault detection coverage [14] may be taken into account after the reliability model is obtained.
- Checkpointing is possible anywhere in the task with a constant checkpoint interval [5]. In practice, it might be difficult to accomplish. But the result of analysis with equidistant checkpointing can give an insight into how checkpoints should be inserted to improve the reliability of a system.
- Checkpointing and recovery overheads are same and remain constant [5].

2 Reliability Analysis

If N_c checkpoints are inserted uniformly into the task whose worst case execution time, relative deadline, slack time, and checkpointing/recovery overhead are T_e, T_d, T_s, and T_c, respectively, as shown in Fig. 1(a), the task is divided into N_c subsections creating N_c time-slots as illustrated in Fig. 1(b). Each time-slot is composed of a part of normal execution and a checkpointing operation, and its length is $\frac{T_e}{N_c} + T_c$. Due to the checkpointing overheads, the available slack time of the task is reduced from T_s to $T_s - N_c T_c$.

With on-line fault detection mechanisms, the recovery process can be performed immediately after a fault occurrence as shown in Fig. 2. To tolerate transient faults that may occur during either checkpointing or recovery operation, at least two secure storage places should be provided and used for checkpointing alternately. The storage place where the earlier state was saved should be used for the current checkpointing operation and the other one where the more recent state was saved should be reserved for a recovery operation in case of a fault occurrence during the current checkpointing operation.

Transient faults can be classified into two types: One type is those which may occur during normal task execution or during checkpointing operations (fault type 'E' and 'C' in Fig. 2), and the other type is those which may occur during recovery operations (fault type 'R' in Fig. 2). Transient faults of the former type

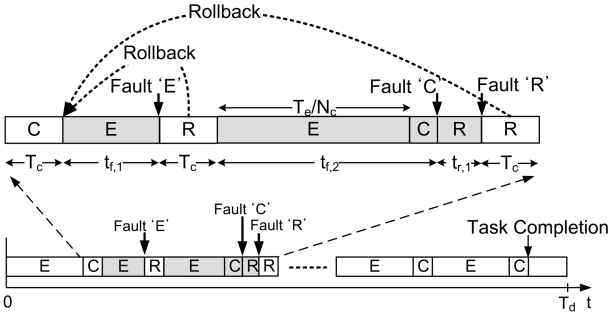


Fig. 2. Checkpointing with on-line fault detection

will be counted to N_f and transient faults of the latter type will be counted to N_r. Then the sum of N_f and N_r is the total number of transient faults that may occur in $[0,\ T_d]$. As shown in Fig. 2, let the time elapsed since the last checkpointing when a fault occurs during normal execution or checkpointing operations be denoted by $t_{f,i}$, $i = 1, 2, \cdots, N_f$, and the time elapsed since the beginning of a recovery operation when a fault occurs during the recovery operation be denoted by $t_{r,l}$, $l = 1, 2, \cdots, N_r$ and define T_t as $\frac{T_e}{N_c} + T_c$. Then the time intervals $t_{f,i}$ and $t_{r,l}$ can be thought of as independent identically distributed (i.i.d.) random variables with exponential distributions on intervals $[0,\ T_t]$ and $[0,\ T_c]$, respectively.

Now we derive the probability of task completion in the presence of transient faults. Since theoretically the number of faults of type 'E' or 'C' in interval $[0,\ T_d]$ ranges from 0 to ∞, N_f ranges from 0 to ∞ as well. A fault of type 'R' can be brought about only after a fault of type 'E' or 'C' occurs. Therefore N_r can range from 0 to N_f. The probability, $P(N_c)$, that the task completes its execution in the presence of transient faults is

$$P(N_c) = Pr\{\text{success with no fault}\} + \sum_{N_f=1}^{\infty} \sum_{N_r=0}^{N_f} Pr\{\text{success with } N_f \text{ and } N_r \text{ faults}\}\,.$$

Since we assumed that faults occur in accordance with a Poisson process, the probability of successful task completion with no fault is

$$Pr\{\text{success with no fault}\} = e^{-\lambda(T_e + N_c T_c)}\,. \tag{1}$$

The probability of successful task completion with N_f and N_r faults is the product of three probabilities: the probability that no fault occurs in N_c time-slots for task completion, the probability that N_f and N_r faults occur before the task completes its execution, and the probability that the time lost by these faults is small enough for the task to meet its deadline.

$$Pr\{\text{success with } N_f \text{ and } N_r \text{ faults}\} = Pr\{\text{no fault in } N_c \text{ time-slots}\} \cdot Pr\{\text{lost time by } N_f \text{ and } N_r \text{ faults is small}\} \cdot Pr\{N_f \text{ and } N_r \text{ faults occur}\}\,. \quad (2)$$

The first probability in the right side of (2) is the same as the probability in (1). In the following, we derive the second and the third probabilities in the right side of (2).

In order for the task to meet its deadline, N_c checkpointing operations, N_f recovery operations should be done within the slack time T_s in spite of the loss in time caused by the faults. If we define T_{ns} as $T_s - N_c T_c - N_f T_c$ and S_{fr} as $\sum_{i=1}^{N_f} t_{f,i} + \sum_{l=1}^{N_r} t_{r,l}$, the probability that the lost time by these faults is small enough for the task to meet its deadline can be expressed as

$$Pr\{\text{lost time by } N_f \text{ and } N_r \text{ faults is small}\} = P(S_{fr} \le T_{ns})\,. \quad (3)$$

In order to get the probability in (3), we have to find out the probability density function (pdf) of S_{fr}. It is already known that the sum of exponential random variables has the Erlang distribution [17]. If T_i's, $i = 1, 2, \cdots, n$, denote i.i.d. exponential random variables, and S_n denotes the sum of these exponential random variables, i.e., $S_n = T_1 + T_2 + \cdots + T_n$, the corresponding pdf's are, respectively,

$$f_{T_i}(x) = \lambda e^{-\lambda x},\ x \ge 0 \quad (4)$$

and

$$f_{S_n}(x) = \frac{(\lambda x)^{n-1}}{(n-1)!}\lambda e^{-\lambda x},\ x \ge 0\,. \quad (5)$$

From the definition of the characteristic function of a random variable [17], the characteristic functions of T_i and S_n are, respectively,

$$\Phi_{T_i}(\omega) = \frac{\lambda}{\lambda - j\omega} \quad (6)$$

and

$$\Phi_{S_n}(\omega) = \left(\frac{\lambda}{\lambda - j\omega}\right)^n\,. \quad (7)$$

The characteristic function of S_{fr} can be derived from those of $t_{f,i}$ and $t_{r,l}$. Since the time intervals $t_{f,i}$ and $t_{r,l}$ are random variables with exponential distribution on intervals $[0,\ T_t]$ and $[0,\ T_c]$, their pdf's are, respectively,

$$f_{t_{f,i}}(x) = \begin{cases} \frac{\lambda e^{-\lambda x}}{1-e^{-\lambda T_t}}, & \text{when } 0 < x < T_t \\ 0, & \text{otherwise} \end{cases}$$

and

$$f_{t_{r,l}}(x) = \begin{cases} \frac{\lambda e^{-\lambda x}}{1-e^{-\lambda T_c}}, & \text{when } 0 < x < T_c \\ 0, & \text{otherwise}\,. \end{cases}$$

Then the characteristic functions of $t_{f,i}$ and $t_{r,l}$ are, respectively,

$$\Phi_{t_{f,i}}(\omega) = \frac{\lambda}{\lambda - j\omega} \frac{1 - e^{-\lambda T_t} e^{jT_t\omega}}{1 - e^{-\lambda T_t}}$$

and

$$\Phi_{t_{r,l}}(\omega) = \frac{\lambda}{\lambda - j\omega} \frac{1 - e^{-\lambda T_c} e^{jT_c\omega}}{1 - e^{-\lambda T_c}} \,.$$

Since the characteristic function of the sum of two random variables is the product of the characteristic functions of these random variables, the characteristic function of S_{fr} is

$$\Phi_{S_{fr}}(\omega) = \left(\Phi_{t_f}(\omega)\right)^{N_f} \left(\Phi_{t_r}(\omega)\right)^{N_r} \,,$$

which is

$$\begin{aligned}
\Phi_{S_{fr}}(\omega) &= \left(\frac{\lambda}{\lambda - j\omega}\right)^{N_f+N_r} \frac{1}{(1 - e^{-\lambda T_t})^{N_f}} \frac{1}{(1 - e^{-\lambda T_c})^{N_r}} \\
&\quad \cdot \sum_{i=0}^{N_f} \binom{N_f}{i} \left(-e^{-\lambda T_t}\right)^i e^{jT_t\omega i} \cdot \sum_{l=0}^{N_r} \binom{N_r}{l} \left(-e^{-\lambda T_c}\right)^l e^{jT_c\omega l} \\
&= \left(\frac{\lambda}{\lambda - j\omega}\right)^{N_f+N_r} \frac{1}{(1 - e^{-\lambda T_t})^{N_f} (1 - e^{-\lambda T_c})^{N_r}} \\
&\quad \cdot \sum_{i=0}^{N_f} \left[\binom{N_f}{i} \left(-e^{-\lambda T_t}\right)^i \sum_{l=0}^{N_r} \binom{N_r}{l} \left(-e^{-\lambda T_c}\right)^l e^{j(iT_t + lT_c)\omega} \right] .
\end{aligned}$$

Then the pdf of S_{fr} can be derived by using the relationship among (4), (5), (6) and (7) as

$$\begin{aligned}
f_{S_{fr}}(x) &= \frac{1}{(1 - e^{-\lambda T_t})^{N_f} (1 - e^{-\lambda T_c})^{N_r}} \\
&\quad \cdot \sum_{i=0}^{N_f} \left[\binom{N_f}{i} \left(-e^{-\lambda T_t}\right)^i \sum_{l=0}^{N_r} \binom{N_r}{l} \left(-e^{-\lambda T_c}\right)^l f(x - iT_t - lT_c) \right] ,
\end{aligned}$$

where

$$f(x) = \frac{(\lambda x)^{N_f+N_r-1}}{(N_f + N_r - 1)!} \, \lambda e^{-\lambda x}, \; x \geq 0 \,.$$

Now that the pdf of S_{fr} is obtained, the probability in (3) can be calculated.

Finally, the third probability in the right side of (2) is derived, which can be decomposed as

$$\begin{aligned}
Pr\{N_f \text{ and } N_r \text{ faults occur}\} &= Pr\{N_f \text{ faults occur}\} \\
&\quad \cdot \; Pr\{N_r \text{ faults occur} \,|\, N_f \text{ faults occur}\} \,.
\end{aligned}$$

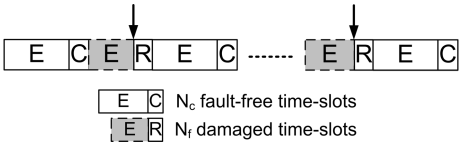


Fig. 3. Possible cases for N_f faults

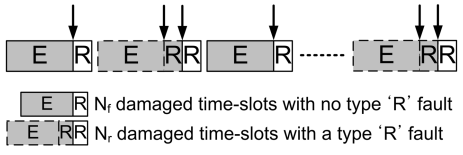


Fig. 4. Possible cases for N_r faults

The probability that a fault occurs and damages a time-slot of length T_t is $1 - e^{-\lambda T_t}$. Among N_c fault-free time-slots, the N_f faults of type 'E' or 'C' can occur in $\binom{N_c+N_f-1}{N_f}$ ways as illustrated in Fig. 3. Therefore the probability that N_f faults occur before the task completion is

$$Pr\{N_f \text{ faults occur}\} = \binom{N_c + N_f - 1}{N_f} \left(1 - e^{-\lambda T_t}\right)^{N_f} .$$

The N_f faults of type 'E' or 'C' would bring about N_f times of recovery operations. Among the N_f recovery operations, N_r operations are damaged by the N_r faults of type 'R'. The probability that N_r recovery operations are damaged is $(1 - e^{-\lambda T_c})^{N_r}$, and the probability that $N_f - N_r$ recovery operations are performed normally is $e^{-\lambda(N_f - N_r)T_c}$. Since the N_r faults may occur among the N_f damaged time-slots in $\binom{N_f}{N_r}$ ways as shown in Fig. 4, the probability that N_r faults of type 'R' occur given that the N_f faults of type 'E' or 'C' have occurred is

$$Pr\{N_r \text{ faults occur} \mid N_f \text{ faults occur}\} = \binom{N_f}{N_r} \left(1 - e^{-\lambda T_c}\right)^{N_r} e^{-\lambda(N_f - N_r)T_c} .$$

Consequently the probability, $P(N_c)$, that the task completes its execution even in the presence of transient faults is

$$P(N_c) = e^{-\lambda(T_e + N_c T_c)} \left(1 + \sum_{N_f=1}^{\infty} \sum_{N_r=0}^{N_f} P_{fr}(N_c, N_f, N_r)\right) ,$$

where

$$P_{fr}(N_c, N_f, N_r) = \binom{N_c + N_f - 1}{N_f} \binom{N_f}{N_r} e^{-\lambda(N_f - N_r)T_c}$$

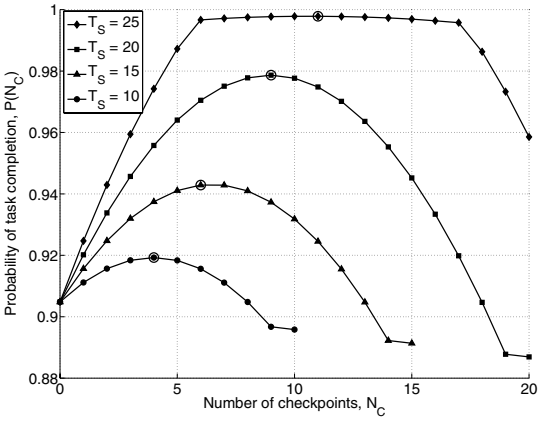


Fig. 5. $P(N_c)$ when $\lambda = 0.001$, $T_e = 100$ and $T_c = 1$

$$\cdot \sum_{i=0}^{N_f} \left[\binom{N_f}{i} \left(-e^{-\lambda T_t} \right)^i \sum_{l=0}^{N_r} \binom{N_r}{l} \left(-e^{-\lambda T_c} \right)^l \int_0^{T_{ns}} f(x - iT_t - lT_c)\, dx \right]$$

and

$$f(x) = \frac{(\lambda x)^{N_f + N_r - 1}}{(N_f + N_r - 1)!} \lambda e^{-\lambda x},\ x \geq 0\,.$$

3 Numerical Examples

Figures 5 and 6 show the graphs of $P(N_c)$ with different values of T_s and λ, respectively. In theses figures, only the terms up to $N_f = 5$ were calculated for each value of $P(N_c)$. The probability of task completion in $[0,\ T_d]$ increases as the available slack time increases. And excessive checkpointing can result in adverse effect. This is because transient faults may occur during checkpointing operations, and more checkpoints than necessary may expose the task to more transient faults and cause the net slack time to decrease leading to the lack of extra time for recovery operations in case of fault occurrences.

4 Best Checkpointing Strategy

From Figs. 5 and 6, it is apparent that there exists the best number of checkpoints which maximizes the probability of successful task completion for the given parameters, such as execution time, checkpointing overhead and available slack time. In practice, the value of λT_d is much smaller than 1. Therefore the value of $P(N_c)$ is mainly dominated by the term resulting from $N_f = 1$ and $N_r = 0$, and can be approximated as

$$P(N_c) \approx e^{-\lambda(T_e + T_c N_c)} \left[1 + N_c \left(1 - e^{-\lambda(T_s - T_c - T_c N_c)} \right) e^{-\lambda T_c} \right]\,.$$

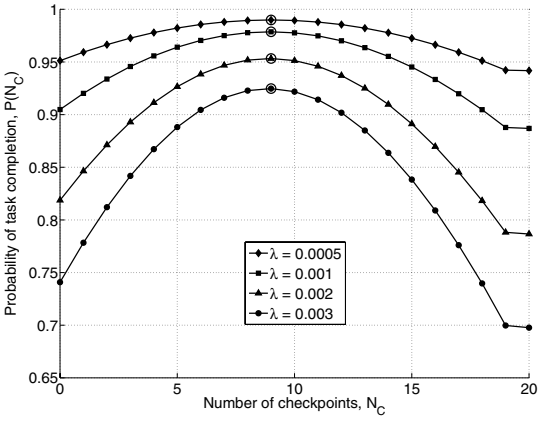


Fig. 6. $P(N_c)$ when $T_e = 100$, $T_c = 1$ and $T_s = 20$

Since it is reasonable that the total checkpointing overhead, $N_c T_c$, is less than the execution time of the task, we have $e^{-\lambda T_c N_c} \approx 1 - \lambda T_c N_c + \frac{1}{2}(\lambda T_c N_c)^2$. And the probability $P(N_c)$ can be approximated once more as

$$P(N_c) \approx e^{-\lambda T_e} \left[1 + (b - a - bc)N_c + (\frac{a^2}{2} - ab)N_c^2 + \frac{a^2 b}{2} N_c^3 \right], \tag{8}$$

where $a = \lambda T_c$, $b = e^{-\lambda T_c}$ and $c = e^{-\lambda(T_s - T_c))}$.

Then, by temporarily assuming N_c to be continuous and by solving the equation which results from taking the derivative of the right side of (8) with respect to N_c and letting it be zero, we can obtain two candidates for the value of N_c that maximizes $P(N_c)$:

$$\left\lfloor \frac{(2b - a) - \sqrt{(2b - a)^2 - 6b(b - a - bc)}}{3ab} \right\rfloor$$

and

$$\left\lceil \frac{(2b - a) - \sqrt{(2b - a)^2 - 6b(b - a - bc)}}{3ab} \right\rceil .$$

The best number of checkpoints is one of the above candidates which leads to the larger value of $P(N_c)$. In Figs. 5 and 6, the points corresponding to the best number of checkpoints for each case are marked by a circle.

5 Conclusion

In this paper, we considered the best equidistant checkpointing strategy for real-time systems from a reliability point of view with the assumption that transient

faults are detected with no latency by on-line detection mechanisms. The reliability analysis shows that the reliability of the system can be improved as much as expected by providing the required slack time with the tasks of the system, and the best number of checkpoints hardly depends on the occurrence rate of transient faults.

References

1. E. Dupont, M. Nicolaidis and P. Rohr, "Embedded Robustness IPs for Transient-Error-Free ICs," IEEE Design & Test of Computers, vol. 19, pp. 56–70, May–Jun. 2002.
2. K. G. Shin, T.-H. Lin and Y.-H. Lee, "Optimal Checkpointing of Real-Time Tasks," IEEE Trans. Computers, vol. C-36, no. 11, pp. 1328–1341, Nov. 1987.
3. S. Punnekkat, A. Burns and R. Davis, "Analysis of Checkpointing for Real-Time Systems," Real-Time Systems, vol. 20, no. 1, pp. 83–102, Jan. 2001.
4. Y. Zhang and K. Chakrabarty, "Energy-Aware Adaptive Checkpointing in Embedded Real-Time Systems," Proc. Design, Automation and Test in Europe Conference and Exhibition, pp. 918–923, Messe Munich, Germany, 2003.
5. S. W. Kwak, B. K. Kim and B. J. Choi, "An Optimal Checkpointing-Strategy for Real-Time Control Systems under Transient Faults," IEEE Trans. Reliability, vol. 50, no. 3, pp. 293–301, Sep. 2001.
6. A. Ranganathan and S. J. Upadhyaya, "Simulation Analysis of a Dynamic Checkpointing Strategy for Real-Time Systems," Proc. 27th Annual Simulation Symp., pp. 181–187, Apr. 1994.
7. D. P. Siewiorek, Reliable Computer Systems: Design and Evaluation, A K Peters, 1998.
8. A. Duda, "The Effects of Checkpointing on Program Execution Time," Information Processing Letters, vol. 16, pp. 221–229, Jun. 1983.
9. P. L'ecuyer and J. Malenfant, "Computing Optimal Checkpointing Strategies for Rollback and Recovery Systems," IEEE Trans. Computers, vol. 37, no. 4, pp. 491–496, Apr. 1988.
10. V. Grassi, L. Donatiello and S. Tucci, "On the Optimal Checkpointing of Critical Tasks and Transaction-Oriented Systems," IEEE Trans. Software Engineering, vol. 18, no. 1, pp. 72–77, Jan. 1992.
11. A. Ziv and J. Bruck, "An On-Line Algorithm for Checkpoint Placement," IEEE Trans. Computers, vol. 46, no. 9, pp. 976–985, Sep. 1997.
12. R. Geist, R. Reynolds and J. Westall, "Selection of a Checkpoint Interval in a Critical-Task Environment," IEEE Trans. Reliability, vol. 37, no. 4, pp. 395–400, Nov. 1988.
13. A. M. Saleh and J. H. Patel, "Transient-Fault Analysis for Retry Techniques," IEEE Trans. Reliability, vol. 37, no. 3, pp. 323–330, Aug. 1988.
14. B. W. Johnson, Design and Analysis of Fault-Tolerant Digital Systems, Addison-Wesley, 1989.
15. J. Sosnowski, "Transient Fault Tolerance in Digital Systems," IEEE Micro, vol. 14, no. 1, pp. 24–35, Feb. 1994.
16. M. Pflanz and H. T. Vierhaus, "Online Check and Recovery Techniques for Dependable Embedded Processors," IEEE Mirco, vol. 21, no. 5, pp. 24–40, Sep.–Oct. 2001.
17. A. Leon-Garcia, Probability and Random Processes for Electrical Engineering, Addison Wesley, 1994.

Parallelizing Serializable Transactions Within Distributed Real-Time Database Systems

Subhash Bhalla and Masaki Hasegawa

The University of Aizu, Aizu-Wakamatsu,
Fukushima PO 965-8580, Japan
{bhalla, d8041201}@u-aizu.ac.jp

Abstract. A real-time database system supports a mix of transactions. These include the real-time transactions that require completion by a given deadline. At the support side, existing concurrency control procedures introduce delays due to non-availability of data resources. The present study makes an effort to introduce a higher level of parallelism for execution of real-time transactions. It considers simple extensions within a transaction processing system. These permit a real-time transaction to avoid delays due to ordinary transactions. These also eliminate elements of other unpredictable delays, due to deadlocks or simple waiting for data resources. Thus, the investigated procedures can perform critical functions in parallel to process time-critical transactions. In effect, it is a model of transaction execution that permits execution of real-time transactions without interference from other executing transactions, and by reducing other probabilistic delays.

1 Introduction

The use of real-time database systems is growing in many application areas such as, industrial process control systems, multi-media systems and many critical data access applications. In such a system, a critical transaction (computational task) is characterized by its computation time and a completion deadline. These systems are characterized by stringent deadlines, and high reliability requirements. Many approaches for implementation of Real-Time systems are being studied [9], [1], [4]. In most cases, the designers tend to extend the available approaches for concurrency control for the new environments. However, we propose to eliminate the elements that cause delays and consider introduction of parallelism based on alternative procedures.

Our research is aimed at isolation of real-time transactions (RTTs). Such transactions are proposed to be executed in parallel with no interference from other transactions. Earlier research efforts within concurrency control also try to isolate transaction classes. For example, isolation of read-only transactions [1], [7], multi-class queries [8], and class for restricting admission [1]. Briefly, our goal is -

1. Isolate a RTT and permit it to execute freely in parallel; and
2. Execute two conflicting RTTs with better completion guarantee for both transactions.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 203–213, 2005.
© IFIP International Federation for Information Processing 2005

For our purpose, we examine the process of data allocation. The characteristics of the 2 Phase Locking based Concurrency Control scheme has been studied. The rest of this paper is organized in the following manner. Section 2. describes the problem with background. Section 3 proposes a validation based selective concurrency control mechanisms that can eliminate delays for the RTTs and reduce interference among other transactions. Section 4 presents a system model of transaction execution. Section 5 examines a criterion for serializability based on the notions of local access graphs (LAGs). Section 6 presents an algorithm for constructing the LAGs. The proof of correctness has been studied in section 7. Section 8 presents a study of performance evaluation. Finally, conclusions and summary are presented in section 9.

2 Nature of Delays

The present study aims at exploring,

- **Transaction Classification**
  - possibility of executing RTTs (time-critical transactions) with no interference from ordinary transactions;
- **Conflicts Among Remaining RTTs**
  - possibility of switching precedence in favor of a more urgent transaction; and
  - eliminate computational losses due to deadlocks, aborts, repeated rollbacks, and excessive overheads associated with access of frequently sought data items.

2.1 Real-Time System Environment

A requirement often imposed on transaction processing systems is that the schedule formed by the transactions, be Serializable [1]. A common method of enforcing serializability is based on two-phase locking. In a real-time environment, preceding ordinary transactions render a substantial portion of database inaccessible to an arriving real-time transaction. The RTDBS environment has the following type of transactions :

1. Real-Time Transactions (RTTs);
   These transactions are characterized by a known computation time estimate (Ct), and a execution time deadline (Dt), as -

$$Ct \ + \ delays \ \ll \ Dt$$

2. Status queries : the read only transactions; and
3. Ordinary Transactions (OTs) : the non-critical transactions that have no execution deadline associated with them.

The prominent delays on account of scheduled transactions are -

1. **Preceding Ordinary Transactions** - The transaction T4 fails to get data items requested (with exclusive access). Also, the transaction T6 fails to get data items requested (with shared access). Thus, an incoming RTT may need to wait until the executing transactions, release the data item [1].
2. **Delays and Conflicts Among RTTs** - The problem is further aggravated in case of RTTs that do not have predeclared read-set and write-set items. These seek locks for data items as these proceed with the computation. Consider a transaction T that reads data item x and writes on data item y. After a delay of d1 units, on receiving a lock for data item x, it computes the update values and seeks an exclusive lock for data item y, which becomes available after a delay of d2 units of time. Hence, the worst time estimate for computation time T is given by :

$$Ct \ + \ (d1 \ + \ d2) \ \ll \ Dt$$

3. **Deadlocks Among RTTs** The situation is further complicated by the occurrence of deadlocks, that can introduce more delays.

Thus, RTTs can have worst case estimates depending on the number of transactions executing in the system and the number of items sought by the transactions. This possibility can lead to a failure of an RTT.

3 The Proposed Model

3.1 Interference from Ordinary Transactions

In many of cases, a large component of the data resources are held by the scheduled (executing) ordinary transactions. The existing approaches are primitive in nature. These make a transaction manager (TM) dependent on multiple data managers (DMs) for priority based executions and aborts [1].

A conceptual model of transaction processing is shown in Figure 2. As per the idea, only the RTTs are permitted to lock data items. The ordinary transactions can execute by performing an additional validation based check [1], [3] [2]. It ensures that, there is no overlap among items in the RTT lock table, and the items read by ordinary transactions. In case of an overlap, the ordinary transactions roll-back. The resulting transaction processing system can perform all RTTs by ignoring existence of ordinary transactions.

For sake of implementation, in case of conflict between a RTT and an OT, the locks granted to ordinary transactions are ignored. The ordinary transaction is informed about possibility of failure during validation. The transaction commitment by the data manager is denied success, in case of a failure (of validation). Such an implementation reduces the data conflicts. These remain to exist among the few executing RTTs.

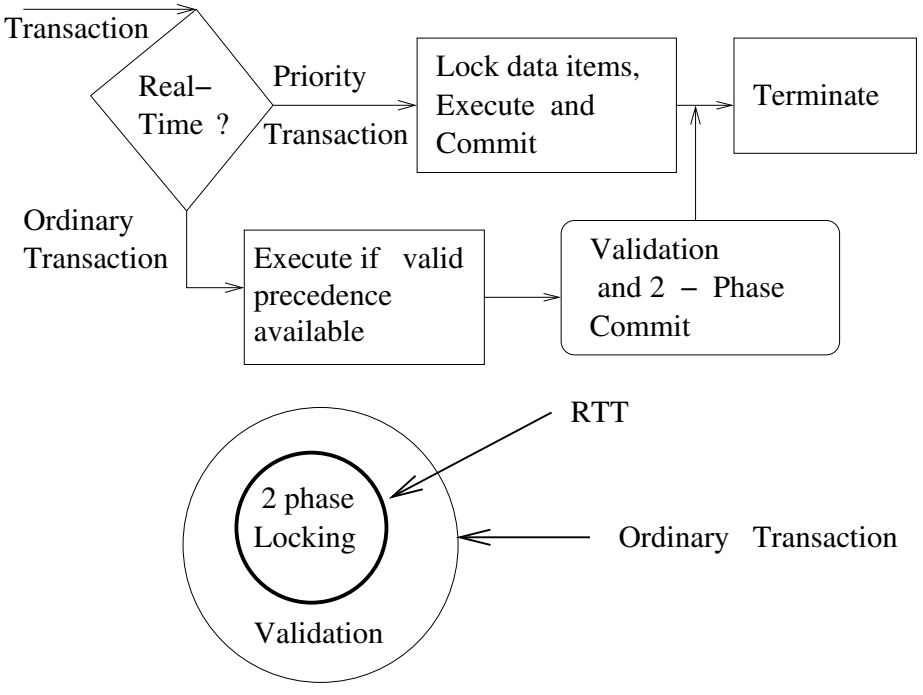


Fig. 1. Execution of real-time transactions in isolation (Brute-force Method)

3.2 Delays and Conflicts Among RTTs

In case of blocking, transactions begin to prepare Transaction-Wait-For graphs (TWFGs). Alternatively, probe messages are sent in order to detect the cause of blocking [1]. Consider an example, a frequently sought after item 'X'. Transactions T4, T3, T5, and T1 arrive in the same order. Other than T4, the three transactions detect that they need to wait for T4. Depending on there relative priority of execution, T3 may abort T4. Subsequently, T3 is aborted by T1, due to a higher priority. Similar aborts in the case of multiple frequently sought data items (**hot-spots** [1]), can cause delays due to deadlocks and frequent aborts. The proposed technique transforms the lock table at the DM level into a local access graph. It aligns the incoming transactions as T4 - T1, T3, T5 (if T4 is completing 2nd phase of '2-phase commit'), or as, T1 - T3, T4, T5.

3.3 Delays Due Dead-Locks

A distributed deadlock occurs when a transaction waits for locks held by another transaction, which in turn, is waiting (directly or indirectly) for locks held by the first transaction. The presence of any deadlock introduces delays and degrades the database performance [5].

4 The System Model

Based on the models of 2 phase locking and real-time computational environment [6], a set of assumptions are organized. It is assumed that a 2 phase locking discipline is followed. In the following section, a scheme to execute transactions in accordance with a precedence order is described.

4.1 Definitions for Real-Time Database System

The distributed database system (DDBS) consists of a set of data items (say set 'D'). The DDBS is assumed to be based on a collection of (fixed) servers that are occasionally accessed by real-time hosts. A data item is the smallest accessible unit of data. It may be a file, a record, an array, or an object. Each data item is stored at a site (one only). However, this assumption does not restrict the algorithm in any way and can be relaxed in a generalized case. Transactions are identified as $T_i, T_j, \ldots$; and sites are represented by $S_k, S_l \ \ldots$; where, $i, j, k, l \ \ldots$ are integer values. The data items are stored at database sites connected by a computer network.

Each site supports a transaction manager (TM) and a data manager (DM). The TM supervises the execution of the transactions. The DMs manage individual databases. The network is assumed to detect failures, as and when these occur. When a site fails, it simply stops running and other sites detect this fact. The communication medium is assumed to provide the facility of message transfer between sites. A site always hands over a message to the communication medium, which delivers it to the destination site in finite time. For any pair of sites S_i and S_j , the communication medium always delivers the messages to S_j in the same order in which they were handed to the medium by S_i.

4.2 The Transaction Model

We define a transaction as a set of atomic operations on data items. The system contains a mixture of real-time transactions and ordinary transactions. We assume that the ordinary transactions can be aborted, in case of a data conflict with the real-time transactions. An operation is either a read (returns the value of the data item), or a write (updates the item with a specified new value). For sake of simplicity, we assume that every transaction can read and write on any data item at most once. For any T_i and data item X, $r_i[X]$ denotes a read executed by T_i on X. Similarly, $w_i[X]$ denotes a write executed by T_i on X. The notation o_i denotes an operation of transaction T_i (i.e., either r_i or w_i). We let OS_i denote the set of all operations in T_i (i.e., $OS_i = \cup_j O_{ij}$). Where the notation, $O_{ij}[X]$ or O_{ij} denotes j-th operation o_i of transaction T_i on a data item X.

We denote by N_i the termination condition for T_i, where $N_i \in \{abort, commit\}$. In general, a transaction does not have to be a totally ordered sequence. When two operations are not ordered relative to each other, these can

be executed in any order. However a read and a write on the same element, must be ordered.

Definition 1 : Two operations $o_i[X]$ and $o_j[X]$ conflict with each other, if they operate on data item (X) and at least one of them is a Write.

Definition 2 : A transaction T_i is a partial order $T_i = \{\sum_i, \ll_i\}$, where

1. $\sum_i = OS_i \cup \{N_i\}$.
2. For any two operations $O_{ij}, O_{ik} \in OS_i$, if $O_{ij} = r(X)$ and $O_{ik} = w(X)$ for any data item X, then either $O_{ij} \ll_i O_{ik}$ or $O_{ik} \ll_i O_{ij}$.
3. $\forall\ O_{ij} \in OS_i,\ O_{ij} \ll_i N_i$.

The items to be locked by the transaction for the purpose of read and write steps are termed as read-set (RS) and write-set (WS), respectively. The union of the read-set and the write-set of a T_i constitutes locking variables (LV_i). Our definition of a transaction conflict is the same as the commonly accepted notion [1]. Hence, $LV_i \cap LV_j = \phi$, implies that no conflict exists between $T_i,\ T_j$.

A transaction asks for an exclusive lock, if it reads and writes on a data item. Non-compatible locks are granted after completion of the preceding transaction. However, the compatible locks are granted in parallel. Local computation starts after the lock grants have been received. After the computation, the write phase is initiated. Updated data items are sent to the data sites and are stored in a temporary memory. It is assumed that, the 'two-phase commit' protocol [1] is employed to guarantee the atomicity of transactions that involve multiple sites. The real-time transactions are accorded priority and are always committed by local DMs, in preference over local computations.

Let $T = T_1, \ldots, T_n$ be a set of active transactions in a DDBS. The notion of correctness of transaction execution is that of serializability [1]. Hence, a transaction number (TN) assigned to a transaction for its identity has a 5 element value, as (site-id,local-clock,type,priority,global-identity).

5 Ordering of Transactions in a Distributed System

In the proposed approach, the transactions are ordered by constructing local access graphs (LAGs) for access requests. In this section, we define a LAG.

Definition 3 : A directed graph G consists of a set of vertices $V = V_1, V_2, \ldots$, a set of edges $E = E_1, E_2, \ldots$, and a mapping function Ψ that maps every edge on to some ordered pair of vertices $< V_i, V_j >$. A pair is ordered, if $< V_i, V_j >$ is different from $< V_j, V_i >$. A vertex is represented by a point, and an edge is represented by a line segment between V_i and V_j with an arrow directed from V_i to V_j.

Insertion of an edge $< V_i, V_j >$ into the graph G =(V,E) results in graph G' =(V',E'), where $V' = V \cup \{V_i, V_j\}$ and $E' = E \cup \{< V_i, V_j >\}$. The union of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is another graph G_3 (written as $G_3 = G_1 \cup G_2$), whose vertex set is $V_3 = V_1 \cup V_2$ and the edge set is $E_3 = E_1 \cup E_3$.

Let, there be a partial ordering relation $\ll_T$ defined over T (the collection of executing transactions $T_1, \ldots, T_n$), to indicate a precedence order among transactions, based on criteria of serializability.

Definition 4 : An access graph of T_i (AG_i) is a graph $AG_i(V, E)$, where $V \subseteq T$, and $E = \{< T_j, T_i > | LV_j \cap LV_i \neq \phi$ and $T_j \ll_T T_i\}$.

Example 2 : Consider the transactions RT_a, RT_b, RT_c, RT_d and RT_e as shown below. Let X,Y,Z be data items. Also,

- $RT_a = r_a(X)r_a(Y)w_a(X)w_a(Y)$.
- $RT_b = r_b(X)r_b(Y)r_b(Z)w_b(X)w_b(Y)w_b(Z)$.
- $RT_c = r_c(Z)w_c(Z)$.
- $RT_d = r_d(X)r_d(Z)w_d(X)w_d(Z)$.
- $RT_e = r_e(Y)r_e(Z)w_e(Y)w_e(Z)$.

Consider a situation, where X,Y and Z are located at one site. The execution of above transactions' operations can follow any one of the sequences in accordance with the criteria of serializability [1]. For each execution (equivalent to a serial execution), the AGs of transactions are different. If we consider the arrival pattern of transactions in the order RT_b, RT_c, RT_d and RT_e, RT_a, then, the corresponding AGs of above transactions are shown in Figure 4. Noting that the real-time transaction priority is highest for RT_a, and lowest for RT_e. In this, $RT_i \rightarrow^x RT_j$ indicates, RT_j is waiting for data item X which will be released after completion of RT_i.

Definition 5 : A local access graph (LAG) of T_i at S_k, is a graph $LAG_{ik}(V, E)$, where, $V \subseteq T$, and $E = \{< T_j, T_i > | LV_{jk} \cap LV_{ik} \neq \phi$ and $T_j \ll_T T_i\}$. In this expression, T_j has previously visited site S_k, and LV_{ik} denotes the part of LV_i, resident at S_k.

When a locking request LR_i (for RT_i) is sent to S_j, a LAG_{ij} is constructed at S_j.

$.RT_a$ z: free	$RT_a \longrightarrow^{x,y} RT_b$	$RT_b \longrightarrow^z RT_c$	$RT_a \longrightarrow^x RT_d$ $RT_b \longrightarrow^{x,z} RT_d$ $RT_c \longrightarrow^z RT_d$	$RT_a \longrightarrow^y RT_e$ $RT_b \longrightarrow^{y,z} RT_e$ $RT_c \longrightarrow^z RT_e$ $RT_d \longrightarrow^z RT_e$
1. AG_a	2. AG_b	3. AG_c	4. AG_d	5. AG_e

Fig. 2. Access Graphs (AGs) of transactions

Observation 1: Let LV_i be stored at sites $S_1, \ldots, S_m$. And, LAG_{ij} be the LAG of T_i at S_j. Then,

$$AG_i = \cup_{j=1}^{m} LAG_{ij}$$

6 An Algorithm to Construct LAG

In this section, we describe some of the terms used in the algorithm. The algorithm is presented in the following section.

6.1 Terms Used to Describe the Algorithm

- Home site ($\mathbf{SH_i}$):
  The site of origin of T_i is referred to as the home site.
- Transaction number ($\mathbf{TN_i}$):
  A unique number (TN_i), is assigned to the transaction T_i on its arrival at the home site. In this paper, both notations TN_i and T_i are used interchangeably and represent individual transactions. A real-time transaction is allotted a distinct identity indicating its type and priority.
- Locking variables ($\mathbf{LV_i}/\mathbf{LV_{ik}}$):
  The items read or to be written by a T_i, constitute the LV_i. The locking variables at S_k constitute LV_{ik}.
- Lock request ($\mathbf{LR_{ik}}$):
  It consists of TN_i and LV_{ik}. It is prepared by SH_i on arrival of T_i, and is sent to each concerned site S_k.
- **Odd edge, Even edge** :
  As per access ordering based on dataflow graphs, an edge $< T_j, T_i >$, such that $TN_j > TN_i$, is called as odd edge. It is treated as a negative priority edge as it needs to be interchanged (verified), upon its occurance.

  The even edges are also called priority edges. A real-time transaction always forms a priority edge (or even edge) with other transactions, due to its real-time priority.
- Access grant status ($\mathbf{AGS_{ik}}$):
  It has values 0 or 1. After granting of all the requested locks of data items at S_k to LR_{ik}, the AGS_{ik} is changed to 1 at site S_k. Otherwise, AGS_{ik} is 0 for waiting transactions.
- Active list:
  The Active list is maintained by each S_k. The Active list of S_k is divided into two tables: active list of lock requests at S_k ($\mathbf{ALT_k}$), and active list of LAGs at S_k ($\mathbf{ALG_k}$). These tables are:
  - $ALT_k = \{(LV_i, AGS_{ik}) \mid T_i$ requested data items at $S_k \}$.
  - ALGk = $\{ LAG_{ik}$ — T_i requested data items at $S_k \}$.

  A transaction T_i is inserted into the ALT_{ik}, after initializing AGS_{ik}. On getting the access grants for LV_{ik}, AGS_{ik} is changed to 1. As a next step, these access grants are sent to SH_i.
- Data table ($\mathbf{DT_i}$):
  This table is maintained at the SH_i for each T_i. The DT_i contains the lock grants (with values) of T_i. Whenever S_k receives any lock grant from another site, it stores it in a corresponding DT_i.

– Status of a transaction ($\mathbf{ST_i}$):
For a T_i, ST_i is maintained at the SH_i. It has values 0 or 1. Initially, ST_i is 0. After receiving all commitment messages in phase 1 of '2-phase commit', ST_i is changed to 1. After this, the final phase of commitment of T_i begins.
– Conflict-set of LAG ($\mathbf{LAG_{ik}.conflict-set}$):
Set of transactions in LAG_{ik} which are in conflict with T_i. That is, $LAG_{ik}.conflict-set = \{vertex-set[LAG_{ik}] - [T_i]\}$.

6.2 Informal Description of the Algorithm

If a transaction needs to access data items, its LR_{ik} are sent to each (concerned) site S_k. The LAG_{ik} is updated at these sites. At any site S_k, if LAG_{ik} contains odd edge $< T_j, T_i >$, then it is an indication of possible blocking or delay. It is proposed that the real-time transactions exchange the precedence with the lock holding transaction, to generate a normal precedence (even edge). In all cases, the odd edge is nullified by exchange of precedence to revoke the grant. This is called the confirmation of the edge.

An **edge is confirmed** by checking the existing AGS_{jk} **locally**, or by sending an "abort the edge" message to the SH_j. That is, if the AGS_{jk} is 0, then an even edge $< T_i, T_j >$ is inserted into the LAG_{jk} and odd edge $< T_j, T_i >$ is deleted from LAG_{ik}. Otherwise, at the SH_j, if T_j is under execution, then a **reverse** grant message is sent to SH_j, and the odd edge $< T_j, T_i >$ is substituted at the LAG_{ik}. The executing transaction T_j performs a partial roll-back.

If T_i is a **real-time transaction**, but the local site is participating in a '2-phase commit' for T_j, the SH_j is sent a message $'T_j.STATE'$. In response, its home site terminates T_j, or communicates the updated values, within a time-out period. Else, the site is treated as a failed site.

If T_i is an **ordinary transaction**, but the local site is executing the 2nd phase of 2 phase locking for T_j, the SH_j is sent a message $'T_j.STATE'$. In response, its home site completes T_j, and communicates the updated values (within a larger time frame). Thus, a time-critical transaction is permitted to execute a revoke grant (if necessary) for the conflicting item, in order to cancel an odd edge.

7 Proof of Correctness

Theorem 1: Let H be the history over T. And there are 'm' sites in the system. Then the execution produced by the algorithm is Serializable.

Proof: In the algorithm, the conflicts are ordered through the LAGs. So, if $G_L = \cup LAG_{ik}$, where i=1 to n, and k= 1 to m, then, we prove the following.
(i) $G_L = SG(H)$.
(ii) G_L is acyclic.
(i) For any two transactions T_i and T_j, if these have a conflict between them, then - either $< T_i, T_j > \in LAG_{jk}$ or $< T_j, T_i > \in LAG_{ik}$ at some S_k. So, the edge set of G_L contains all conflicting pairs. So, $G_L = SG(H)$.

(ii) The algorithm follows two phase locking principle. That is, the data manager does not releases any lock until it acquires all required locks. Then, G_L is acyclic [1].

Theorem 2: Let $T = \{T_1, T_2, \ldots, T_n\}$ be the set of conflicting transactions in the system. The above algorithm results in a deadlock free environment.

Proof: Every deadlock cycle results in the formation of at least one odd edge in some LAG_{ik} at some S_k. Suppose, T_i forms an odd edge $< T_k, T_i > \in LAG_{ip}$ at some S_p. In this case, in accordance with the proposed algorithm, if T_k has not started completion of transaction commitment, the odd edge is removed, and corresponding even edge is inserted in the respective LAG, which breaks the deadlock cycle. Otherwise, if T_k has started completion and broadcast of committed values, then also the cycle will be broken.

Theorem 3: Let $T = \{T_1, T_2, \ldots, T_n\}$ be the set of conflicting transactions in the system. The above algorithm results in a priority inversion free environment.

If all transactions access data items in accordance with the precedence order assigned to them, no priority inversion occurs. Priority inversion results from formation of an odd edge in some LAG_{ik} at some S_k. As shown above, odd edges are eliminated, as these occur.

8 Summary and Conclusions

In the distributed locking based approaches, if transactions from different sites, are in serializability conflict, then some of the submitted transactions are rejected. Transaction rejects and delays are two of the main problems that concern real-time transaction processing activity. In this study, a procedure has been identified that shows a possibility of execution of critical transactions under serializability conditions. A higher level of concurrency is achieved, as a result of removal of excessive blocking, and roll-backs for critical transactions.

References

1. P.A.Bernstein, V.hadzilacos and N.Goodman, "Two Phase Locking", Chapter 3, *Concurrency control and recovery in database systems*, Addison-Wesley,1987.
2. Bhalla, S. and S.E. Madnick, "Asynchronous Backup and Initialization of a Database Server for Replicated Database Systems", The Journal of Supercomputing, Vol. 27, No. 1, pp. 69-89, Kluwer Academic Publishers, January 2004.
3. Reddy, P.K., and S. Bhalla, "Asynchronous Operations in Distributed Concurrency Control", IEEE Transactions on Knowledge and Data Engineering, Vol. 15, No. 3 May-June 2003, pp. 721-733.
4. Bhalla, S., "The Performance of an Efficient Distributed Synchronization and Recovery Algorithm", The Journal of Supercomputing, Kluwer Academic Publishers, vol. 19, no. 2, June 2001, pp. 199-220.

5. A.N.Choudhary, " Cost of distributed deadlock detection: A performance study", *Proceedings of the Sixth International Conference on Data Engineering*, pp.174-181, February 1990.
6. Korth H.F., E. Levy, and A. Silberschatz, "Compensating Transactions: a New Recovery Paradigm," in *Proc. 16th Intl. Conf. Very Large Databases*, Brisbane, Australia, 1990, pp. 95-106.
7. Lam Kwok-wa, S.H. Son, V.C.S. Lee, and Sheung-Lun Hung, "Using Separate Algorithms to Process Read-Only Transactions in Real-Time Systems", *Proceedings of IEEE Real Time Systems Symposium*, Dec. 1998, pp. 50-59.
8. Pang, H., M.J. Carey, and M. Linvy, "Multiclass Query Scheduling in Real-time Database Systems", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 7, No. 4, August 1995.
9. Ramamritham K., "Real-Time Databases", *Distributed and Parallel Databases*, Kluwer Academic Publishers, Boston, USA, Vol. 1, No. 1, 1993.

Timing Analysis of Distributed End-to-End Task Graphs with Model-Checking

Zonghua Gu

Department of Computer Science,
Hong Kong University of Science and Technology

Abstract. Real-time embedded systems must satisfy system-level timing constraints between external sensor inputs and actuator outputs. Real-time scheduling theory can be used to verify that the system is schedulable, that is, no deadlines are missed, but that alone is not enough. Given that the system is schedulable, how to verify that it satisfies system-level end-to-end timing constraints, such as freshness, correlation and separation? To address this question, we adopt the approach of formal modeling and model-checking. Specifically, we use Timed Automata and the model-checker UPPAAL for verification purposes. We have developed generic modeling templates for a class of distributed task systems that can be used as input the the model-checker in order to verify system-level end-to-end timing constraints. We use an application example of distributed real-time control system to illustrate the utility of our approach.

1 Introduction

Real-time embedded systems must satisfy system-level timing constraints between external sensor inputs and actuator outputs. Gerber *et al* [1] identified three classes of *system-level end-to-end timing constraints*:

- *Freshness constraints* bound the maximum time it takes for the data to flow through the system. For example, the time interval between reading input data d from the sensor and writing output data calculated using d to the actuator can not exceed 10ms.
- *Correlation constraints* bound the maximum time-skew between several inputs used to produce an output. For example, if input data d_1 and d_2 are used to produce output data d_3, then d_1 and d_2 must be sampled within 2ms of each other. Generally it is desirable to minimize the data correlation from distributed sensors in order for the controller to have a realistic and consistent view of the physical environment.
- *Separation constraints* bound the maximum jitter between consecutive values on a single output channel.

Given a set of such timing constraints, as well as timing attributes of the tasks composing the system, we would like to verify that the system satisfies

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 214–223, 2005.
© IFIP International Federation for Information Processing 2005

these constraints. Real-time scheduling theory can be used to prove that the system is schedulable, that is, no deadlines are missed, but that alone is not enough. A schedulable system may still violate system-level freshness, correlation and separation constraints. In this paper, we address this question: *Given that the system is schedulable, how to verify that it satisfies system-level end-to-end timing constraints, such as freshness, correlation and separation?*

The growing complexity of modern real-time embedded systems makes it imperative to apply formal analysis techniques at early stages of system development. In this paper, we adopt the approach of formal modeling and model-checking, which is a promising technique for analysis of finite state systems by exhaustive exploration of the system state space. Specifically, we use *Timed Automata* (TA) and the model-checker UPPAAL [2] for this purpose. Our main contribution in this paper is developing modeling techniques for a class of distributed task systems that are used as input the the model-checker UPPAAL to verify system-level end-to-end timing constraints. We make the assumption that the system is already schedulable before applying model-checking. Without this assumption, we would have a much larger state space due to deadline misses, an error condition that should never arise in a production system, without gaining any additional insight into the system's normal operation.

Some authors have developed modeling formalisms and analysis techniques for real-time scheduling analysis. For example, ACSR [3] is a resource-aware real-time process algebra for specification and formal verification of distributed real-time systems. Its main features include the ability to specify resources and their usage by system components, and prioritized execution that allows to express different preemptive and non-preemptive scheduling policies. Similarly, TIMES [4] is based on Timed Automata, and can be used for modelling, schedulability analysis and synthesis of optimal schedules for a set of tasks that are triggered periodically by timers, or sporadically by external interrupts. These approaches have the benefit of being able to deal with more general task models than real-time scheduling theory, which is typically restricted to periodic task models, and has to make pessimistic assumptions when dealing with sporadic or aperiodic tasks. However, modeling real-time scheduling inevitably introduces a much larger state space caused by interleaving of task executions on a shared processor, and has a negative impact on the scalability of model-checking, which is already the most important limiting factor for its industry adoption. In contrast, we adopt an approach of separation of concerns. We do not consider real-time scheduling issues caused by interference among tasks sharing a processor. We either assume a distributed system, where each task has its own dedicated processor, or a system with shared processors, but the timing attributes of tasks already take into account interference caused by multiple tasks sharing one processor, i.e., we use *worst-case response time* instead of *worst-case execution time* as a task's timing attribute. By using real-time scheduling theory to verify the schedulability assumption, and model-checking to verify system-level timing properties, we can achieve much better scalability than using model-checking to check for both schedulability and system-level timing properties.

This paper is structured as follows. In Section 2, we present our system modeling techniques. In Section 3, we use an application example to illustrate the complexity involved in timing analysis of distributed end-to-end task systems, and apply our modeling and model-checking approach to its timing analysis. In Section 4, we discuss related work, and in Section 5, we draw some conclusions.

2 System Modeling with TA

A *Timed Automaton* (TA) is a standard finite-state automaton extended with a finite collection of real-valued clocks, which proceed at the same rate and measure the amount of time that has elapsed since they were last reset. The model-checker UPPAAL [2] adds a few extensions to the standard definition of TA, such as integer variables, CCS-style [5] synchronous communication, urgent channels, etc. For space limitations, we do not provide a detailed description of TA and UPPAAL.

We define the system under analysis with an Asynchronous Task Graph [1], which is a *Directed Acyclic Graph* (DAG), where vertices denote tasks, which have input and output data ports. Directed edges denote dataflow communication between tasks. Tasks execute asynchronously and communicate with each other through shared buffers of size 1. Reading and writing are both non-blocking, i.e., writing a data token to a shared buffer overwrites the previous data token contained in it. Reading and writing are both assumed to be instantaneous, hence we do not require the use of mutual exclusion mechanisms such as mutex or semaphore. Forks and joins in the task graph are of type *AND*. That is, when a task executes, it reads in one data token from each of its input ports in an instantaneous, atomic action at the start of its execution, and then writes out one data token to each of its output ports in another instantaneous, atomic action at the end of its execution.

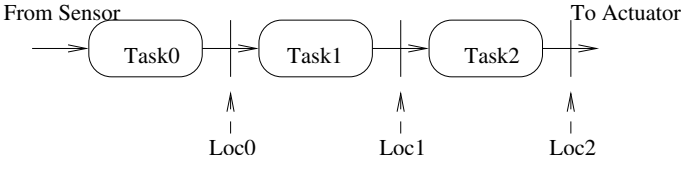


Fig. 1. An end-to-end task graph consisting of a linear chain of sub-tasks, with *atomic, non-interleaving* task execution

Consider a linear chain of sub-tasks in an end-to-end task graph from external stimulus from the environment to eventual output to the environment, as in Figure 1. The taskset is $\{T_0, T_1, \ldots, T_{N-1}\}$, where T_{i-1} is the immediate predecessor of T_i, where $i = 1, \ldots, N-1$. Since tasks communicate through shared buffers, the system events that we are concerned with are the read/write events of the shared buffers. Suppose that we know the execution time intervals of each

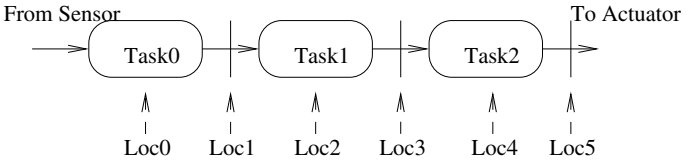


Fig. 2. An end-to-end task graph consisting of a linear chain of sub-tasks, with *non-atomic, interleaving* task execution

task block, and we want to model this task chain with timed automata. This is straightforward if the execution is single-rate, and there is no pipelining, i.e., each invocation of the task chain executes from start to finish before the next invocation. However, the situation becomes more complicated if *pipelined* and *multi-rate executions* are allowed. The execution is pipelined if it is possible for new inputs to come in from the environment before the previous output has been produced. The execution is multi-rate if it is possible for each task's execution rate to be different from each other. Think of each piece of data read in from a sensor task as a data token. In the case of pipelined and multi-rate execution, multiple data tokens read in by the same sensor task in different execution periods may coexist in the system at the same time.

For illustration purposes, imagine that we attach a real-time clock that is reset to 0 with each data token coming in from the head of the pipeline, and mark this particular clock as being in use. Every time a data token goes through the entire pipeline and falls off the end of the task chain, we pick up the clock on that token, read off the value as the end-to-end delay suffered by that token, and mark the clock on that token as free and can be reused for a new token in the future. When the pipeline stalls at a task i, and the data token in the location before task i gets overwritten, We can also mark the clock of that data token as free. We can view a fresh data token read in from the environment as birth of that token, and token falling out of the end of the pipeline, or overwritten in the middle of the pipeline, as death of that token. A clock not currently in use is attached to each newborn token, and is marked in use. Upon death of a data token, either because it has gone through the entire pipeline, or it was over-written (killed prematurely) in the middle of the pipeline, we mark its clock as free and available for future use. It is important to note that when attaching a clock to a newborn token t_{new}, that clock must not be already in use by some other token t_{old} currently in the pipeline, for otherwise we will lose track of the time stamp of t_{old}.

If task execution is atomic and non-interleaving, then we need to keep track of clocks at the the inter-task buffer locations, as shown in Figure 1. Atomicity means that no other task can perform read or write operations in between a task's start and end. However, if task execution is non-atomic and interleaving, which means that it is possible for other tasks to perform read or write operations in between a task's start and end, then we need to keep track of clocks at both the inter-task buffer locations and the task bodies between its start and

end, as shown in Figure 2. In other words, we can view a task that executes non-atomically as two tasks, one read task and one write task, which execute atomically, and view the task body between its read and write operations as another data buffer.

Atomic execution is the case where tasks run on a single processor with non-preemptive scheduling, which effectively serializes the task execution. Non-atomic execution is the case where tasks run on a single processor with preemptive scheduling (pseudo concurrency), or where tasks run on a multi-processor platform (true concurrency).

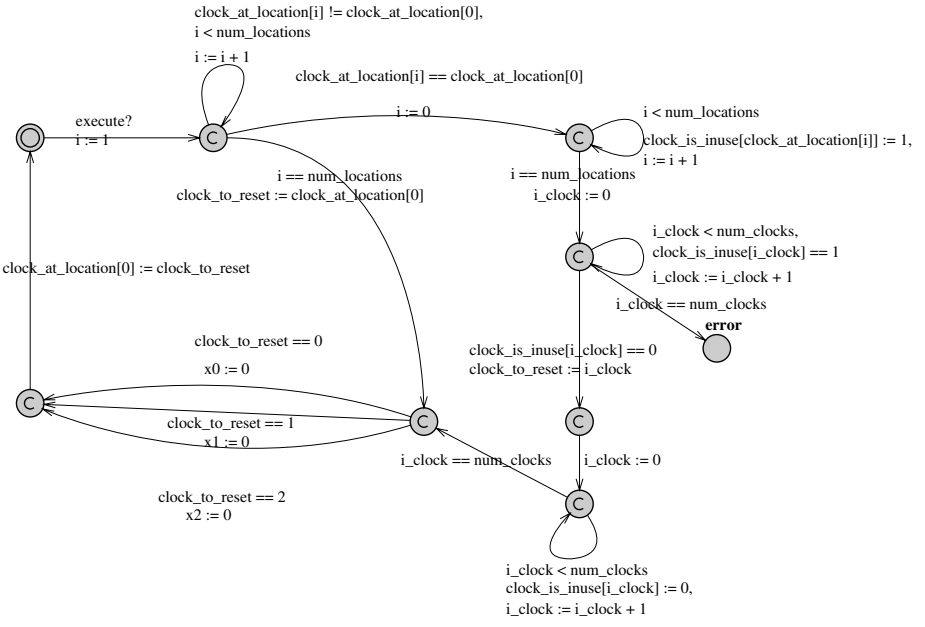


Fig. 3. TA template for tracking the timestamps of $location_0$

The discussion above applies to a linear chain of tasks, but it can be easily extended to deal with a more general task graph with AND style synchronization. When a data token flows to an AND fork in the task graph, the token and its clock are replicated and sent along each of the forked paths. When several data tokens converge at an AND join in the task graph, all the tokens should be carrying the same clock that was replicated at the previous fork to ensure data consistency. Note that this behavior is not necessarily guaranteed. A certain branch of the AND fork may have a pipeline stall, so that the AND join point may see different clocks from different incoming paths. This should be detected and flagged as a runtime error.

To instantiate these concepts within the modeling tool UPPAAL, Figure 3 shows a generic TA template for modeling the output buffer of the head of the

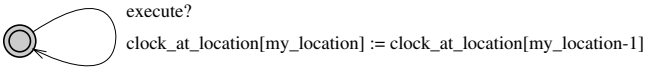
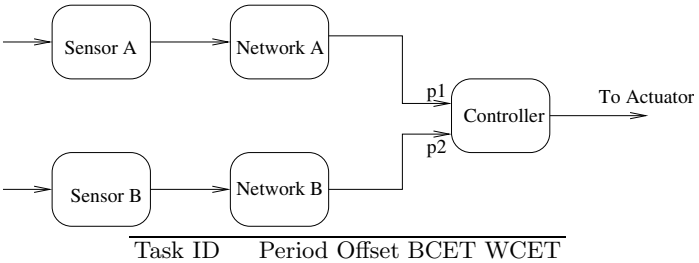


Fig. 4. TA template for tracking the timestamps of locations other than $location_0$

task chain ($location_0$), and Figure 4 shows a generic TA template for modeling of buffers at locations other than $location_0$. The latter template is simple: the task simply copies the clock value at its input port to its output port. The task at the head of the chain needs to read in a fresh data token from the environment and attach to it a clock that is not currently in use. With these model templates, we can easily model larger systems by instantiating the model with different parameters.

3 An Application Example

Consider the system in Figure 5, where two distributed sensors read data from the external environment and feed them into the controller, which processes the sensor data and compute signals to the actuator. The network connections between sensors and controllers are modeled as separate tasks. We assume this is a distributed system, where each task has its own dedicated processor, so we do not need to address real-time scheduling issues caused by multiple tasks sharing one processor. We can view the system as being composed of 2 linear task-chains: Path A is from Sensor A to Network A to Controller, and Path B is from Sensor B to Network B to Controller. Task execution is non-atomic, hence each task chain corresponds to Figure 2. All tasks read input at startup and write output at finish. The table in Figure 5 shows the task timing parameters.



Task ID	Period	Offset	BCET	WCET
$SensorA$	10	0	1	3
$SensorB$	10	1	3	4
$Controller$	10	7	2	2
$NetworkA$	–	1	2	–
$NetworkB$	–	2	5	–

Fig. 5. A system consisting of two distributed sensors and one central controller

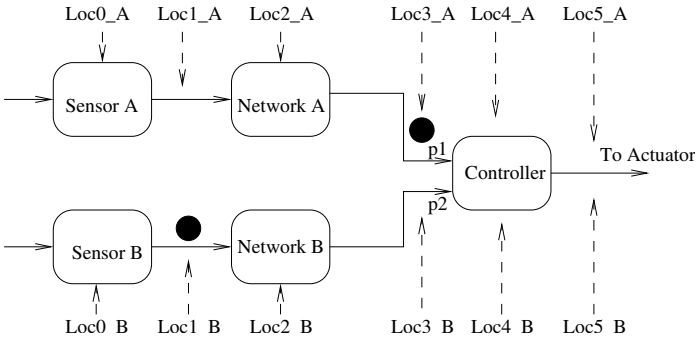


Fig. 6. Each black dot denotes one data token. The figure shows that the input data token read by Sensor A has gone through Network A and reached the input port $p1$ of the Controller, while the input data token read by Sensor B has not gone through Network B yet.

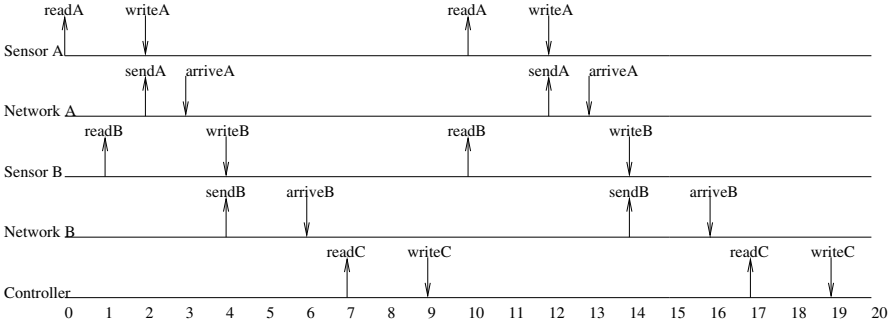


Fig. 7. One possible execution trace for the taskset specified in Figure 5

We have two system-level end-to-end timing constraints:

- *Freshness constraint*: the time interval between reading input data from either Sensor A or Sensor B to writing output data by the Controller must be less than 10ms;
- *Correlation constraint*: if input data d_1 read by Sensor A and input data d_2 read by Sensor B are used to produce output data d_3, then d_1 and d_2 must be sampled within 2ms of each other.

Figure 6 shows a snapshot during system execution. Figure 7 shows one possible execution trace for two consecutive execution periods. Sensor A reads input at time 0 and writes output at time 2ms. Network A is triggered by the completion event of Sensor A, and takes 1 time unit to deliver the message to input port $p1$ of the Controller. Similarly, Sensor B reads input at time 1ms and writes output at time 4ms. Network B is triggered by the completion event of Sensor

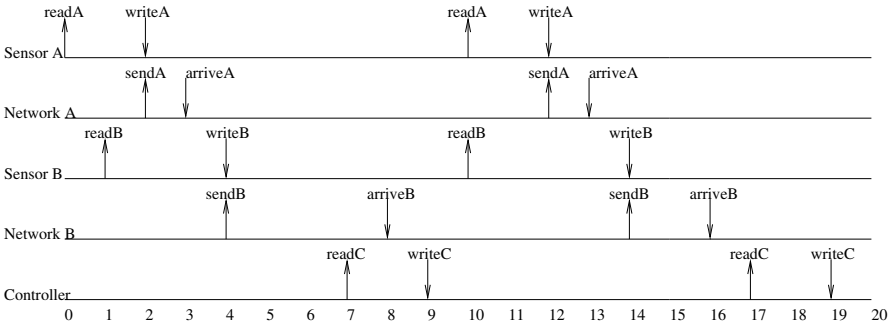


Fig. 8. Another possible execution trace for the taskset specified in Figure 5

B, and takes 2ms to deliver the message to input port $p2$ of the Controller. The Controller reads its input at time 7ms, and writes its output to the actuator at time 9ms. The next period is a repeat of the first. Therefore, the end-to-end delay of data tokens from Sensor A to the external actuator is 9ms, and from Sensor B to the external actuator is 8ms, both within the freshness constraint of 10ms. The correlation between data tokens at the 2 controller input ports is 1ms, i.e., the data token at $p1$ is sampled from Sensor A 1ms earlier than the data token at $p2$ from Sensor B. So the correlation constraint is also satisfied.

However, if Network B causes a slightly longer time delay of 4ms instead of 2ms, but still within the specification of [2,4]ms, then we have a different data token propagation scenario, as shown in Figure 8. Now the end-to-end delay for Sensor B's data token is 18ms, while the delay for Sensor A's data token remains at 9ms. So the freshness constraint is violated for Sensor B's data token. At the time of data reading of the Controller at time 17ms, the token at $p1$ from Sensor A is 7ms old, while the token at $p2$ from Sensor B is 16ms old. So there is a 9ms age difference between the two data tokens read by the Controller. This violates the correlation constraint, which states that the two input data tokens must be sampled within 2ms of each other. Intuitively, the data token read by Sensor A at time 0 was consumed by the Controller at time 7ms before the arrival of data token read by Sensor B at time 1ms, due to the long time delay caused by Network B. Therefore, the data token read by Sensor B at time 1ms is paired up with the data token read by Sensor A at time 10ms in the next execution cycle. This caused the large age difference between the data tokens consumed by the Controller at time 17ms.

As this example indicates, a small additional delay at Network B can cause a relatively large difference for the end-to-end delay and input correlation. Therefore, it is important to analyze the system timing behavior carefully. For realistic systems with larger sizes, it is tedious and error-prone to perform the analysis by hand, therefore, it is desirable to apply automated techniques such as model-checking to discover timing anomalies by exhaustive state space exploration.

We have used UPPAAL to prove the following properties:

- The maximum end-to-end delay for sensor A data is 9ms.
- The maximum end-to-end delay for sensor B data is 18ms.
- the maximum age difference between the data tokens at the two input ports of the Controller is 9ms.

If we change the task parameters, so that Network B task causes a deterministic delay of 2ms instead of an interval of [2,4]ms, we can verify that the maximum age difference reduces to 1ms, and the maximum end-to-end delay of Sensor B data reduces to 8ms. This corresponds to avoiding the execution scenario in Figure 8.

4 Related Work

Wall *et al* [6] designed a TA model for end-to-end task graphs. However, the size of their model grows exponentially with the number of tasks, while the size of our model grows only linearly. We can model larger systems easily by instantiating the proposed modeling templates with different parameters. In general, to keep track of clocks at N buffer locations, the automaton in their approach consists of $N * N$ locations and $N * N * N$ edges. Furthermore, they only considered atomic task executions (non-preemptive scheduling on a single processor), and they used a task execution trace to drive the Time Stamp Tracker automaton. We take into account non-atomic task executions, and accurately model task execution dynamics, which enables exploration of the full system state space with model-checking.

Haveman *et al* [7] manually analyzed a task model that consists of 2 sensors and 1 monitor that merges the 2 sensor readings. They considered a general multi-rate execution scenario, where each task may execute at arbitrary periods that are not necessarily harmonically related, and derived equations for calculating various end-to-end timing properties. We can map their task model into the TA modeling framework discussed in this paper, and analyze it through model-checking, thus obviating the need for tedious and error-prone manual analysis.

There are other techniques and tools for model-checking for timed automata, e.g., Kronos [8], which provides slightly different definitions of timed automata. For example, UPPAAL's definition of TA adopts CCS-style [5] pair-wise synchronization between two automata, while Kronos' definition uses globally-shared synchronization among all automata in the system. Our choice of the model-checker UPPAAL was motivated by several factors, including modeling convenience (pair-wise synchronization turns out to be convenient for our particular problem) and ease of use (UPPAAL provides a graphical user interface, while the other tools only provide textual interfaces). However, the main concepts of our approach are not restricted to any particular tool, but are generally applicable to other formalisms and model-checkers.

5 Conclusions

In this paper, we have explored application of model-checking to timing analysis of distributed end-to-end task graphs. We have developed generic modeling templates for a class of distributed task systems that can be used as input to the the model-checker UPPAAL in order to verify system-level end-to-end timing constraints. We have used a two-sensor-one-controller system to illustrate the complexity involved in end-to-end timing analysis, and applied our techniques to modeling and analysis of this system. Our approach is applicable to any real-time system that satisfies the assumptions stated in Section 2, i.e., tasks executing asynchronously and communicating through shared buffers.

References

1. R. Gerber, S. Hong, and M. Saksena, "Guaranteeing end-to-end timing constraints by calibrating intermediate processes," in *Proc. IEEE Real-Time Systems Symposium*, December 1994, pp. 209–213.
2. J. Bengtsson, K. G. Larsen, F. Larsson, P. Pettersson, and W. Yi, "UPPAAL - a tool suite for automatic verification of real-time systems," in *Proc. Workshop on Verification and Control of Hybrid Systems*, October 1995, pp. 232–243.
3. H. Ben-Abdallah, D. Clarke, I. Lee, and O. Sokolsky, "Paragon: A paradigm for the specification, verification, and testing of real-time systems," in *Proc. IEEE Aerospace Conference*, Feb 1997, pp. 469–488.
4. T. Amnell, E. Fersman, L. Mokrushin, P. Pettersson, and W. Yi, "Times: a tool for schedulability analysis and code generation of real-time systems," in *Proc. International Workshop on Formal Modeling and Analysis of Timed Systems*, Sept. 2003.
5. R. Milner, *A Calculus of Communicating Systems, LNCS 92.* Springer-Verlag, 1980.
6. A. Wall, K. Sandstrom, J. Maki-Turja, C. Norstrom, and W. Yi, "Verifying temporal constraints on data in multi-rate transactions using timed automata," in *Proc. IEEE International Conference on Real-Time Computing Systems and Applications*, December 2000, pp. 263–270.
7. J. Haveman, "Transaction decomposition: refinement of timing constraints," in *Proc. South Pacific Conference on Formal Methods*, 1997.
8. S. Yovine, "Kronos: A verification tool for real-time systems," *Software Tools for Technology Transfer*, vol. 1, pp. 123–133, 1997.

Voronoi-Based Improved Algorithm for Connected Coverage Problem in Wireless Sensor Networks

Jie Jiang, Zhen Song, Heying Zhang, and Wenhua Dou

School of Computer,
National University of Defense Technology,
410073, Changsha, China
jiangjie@nudt.edu.cn

Abstract. In this paper, we consider the connected coverage problem and aim to construct a minimal connected cover set that is sufficient for a given query in wireless sensor networks. We propose a centralized, Voronoi tessellation (CVT) based algorithm to select the minimum number of active sensor nodes needed to cover the target region completely. The constructed sensor set proves to be connected when sensor node's communication range is at least twice of its sensing range. For other situations where the CVT algorithm alone cannot guarantee the network connectivity, we design a Steiner minimum tree (SMT) based algorithm to ensure the network connectivity. Theoretical analysis and simulation results show that our algorithm outperforms the greedy algorithm in terms of both the time complexity and the needed number of sensor nodes that must be kept active to respond to a given query.

1 Introduction

Because of advances in micro-sensors, wireless networking and embedded processing, wireless sensor networks (WSN) are becoming increasingly available for commercial and military applications, such as environmental monitoring, biological attack detection, and battlefield surveillance, etc [1, 2].

Minimizing energy consumption to prolong the network lifetime is a major design objective for wireless sensor networks since tiny sensor devices usually operate on limited battery power due to low cost and small size. Many research efforts have been made to improve the energy efficiency of wireless sensor networks [3, 4, 5, 6, 7].

In this paper we aim to improve the energy efficiency of wireless sensor network by constructing an energy-efficient network topology. The motivation of our work is the spatial query execution in sensor networks. In the process of spatial query, the sink node needs to gather sensor data within a specific target region in order to obtain the status of the monitored target region. An energy efficient way to respond to the query is to activate only a subset of sensor nodes to monitor the field and send back report data. The active node subset must satisfy two requirements: (1) coverage requirement. Each point in the monitored region must

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 224–233, 2005.
© IFIP International Federation for Information Processing 2005

be covered by at least one sensor in this subset. (2) connectivity requirement. The communication network induced by active nodes must be connected.

For a given target region, the problem of finding a minimal subset of sensor nodes which meets the above two requirements at the same time is NP-hard. We propose a centralized Voronoi tessellation (CVT) based heuristic algorithm to compute the near optimal set of sensor nodes needed to cover the target region completely. In case of sensor's communication range is at least twice of its sensing range, the constructed sensor set is connected. In other cases where the CVT algorithm alone cannot guarantee the network connectivity, we design a Steiner minimum tree (SMT) based algorithm to ensure the communication connectivity. The proposed algorithm outperforms the greedy algorithm[9] in terms of both time complexity and the size of active sensor set.

2 Problem Statement

We consider static sensor networks in the 2D plane $\mathbb{R}^2$ and use binary sensing model to model sensor node's sensing capability. In this model, the coverage area of a sensor node s_i is a circle centered at s_i with radius R_s (sensing radius). The circular sensing/coverage area is called s_i's *sensing disk* and denoted by $S_i = \{p \in \mathbb{R}^2 | d(p, s_i) \leq R_s\}$, where d is the Euclidean distance metric. Each senor node has a limited communication range. $N(i) = \{s_j | d(s_j, s_i) \leq R_c, j \neq i\}$ denotes s_i's communication neighbor sets, where R_c is sensor node's wireless communication radius.

Definition 1. *(Connected cover set) A set of sensors S is a cover set for a region R if each point in R is covered at least by one sensor in S. If the communication graph induced by the cover set S is connected, set S is called a connected cover set (CCS) for R.*

For a given sensor network S deployed in a target region R, our goal is to calculate the minimum set of active senor nodes that can both cover the region R completely and form a connected communication network. This problem has been proved to be NP-hard and a greedy algorithm has been proposed in [9]. In this paper, we propose a new, novel algorithm based on Voronoi diagram.

3 Voronoi-Based Heuristic Algorithm

The minimal connected cover set is constructed in two steps. First, we construct the minimal cover set based on the Voronoi tessellation of the target region. When $R_c \geq 2R_s$, the constructed cover set proves to be connected. When $R_c < 2R_s$, we design a Steiner minimum tree (SMT) based algorithm to find the additional sensors needed to make the communication graph induced by the cover set connected. Not loosing generality, we assume that the target region is bounded and convex.

3.1 CVT Algorithm

The minimal cover set can be obtained by turning off the maximal number of redundant sensors. A sensor is redundant if its sensing coverage is completely subsumed by other sensors. Turning off a redundant sensor will not leave sensing hole in the target region.

Definition 2. *(Sensing hole) A sensing hole is a subarea of the target region R where points are not covered by any sensor.*

According to definition 1 and 2, a target region R is completely covered by a sensor network is equal to there is no sensing hole in R. The sensing hole is detected by utilizing the Voronoi tessellation [10] of R. Given a finite set of sensor nodes $S = \{s_1, \ldots, s_n\}$, the Voronoi region V_i associated with node s_i is defined by $V_i = \{p \in \mathbb{R}^2 | d(p, s_i) \leq d(p, s_j), j \neq i\}$, where d denotes the Euclidean distance. In general, the Voronoi polygon associated with node s_i is unbounded if s_i is on the boundary of the convex hull of S [10]. But when R is bounded, the unbounded Voronoi polygons of sensors on the boundary of the convex hull of S will be bounded by the boundary of R.

Definition 3. *(Bounded Voronoi tessellation) A Voronoi tessellation with all Voronoi polygons bounded is called a bounded Voronoi tessellation.*

Let $BVT\,(S, R)$ denote the bounded Voronoi tessellation of the bounded region R when taking S as the generator set, $BVC_i\,(S, R)$ denote the bounded Voronoi polygon associated with sensor s_i, $BVV_i\,(S, R)$ denote the set of vertices of $BVC_i\,(S, R)$ and $BVN_i\,(S, R)$ denote s_i's Voronoi neighbors in $BVT\,(S, R)$. Since the Voronoi polygon is convex [10] and R is also convex, all the Voronoi polygons in the bounded Voronoi tessellation are convex.

Theorem 1. *Consider a sensor network deployed in a bounded, convex region R. R is completely covered by S if and only if for each sensor $s_i \in S$, all the vertices of $BVC_i\,(S, R)$ are within s_i's sensing disk S_i.*

Proof. (Sufficient condition) For any sensor $s_i \in S$, if all the vertices of $BVC_i\,(S, R)$ are within S_i, then $BVC_i(S, R) \subseteq S_i$. Since $R = \bigcup_{s_i \in S} BVC_i(S, R)$, we have $R = \bigcup_{s_i \in S} BVC_i(S, R) \subseteq \bigcup_{s_i \in S} S_i$. Therefore S is a cover set for R.

(Necessary condition) If S is a cover set for R, each point in R is covered at least by one sensor in S. According to the definition of Voronoi tessellation, each point must be covered by its Voronoi generator. ■

Theorem 1 gives a method to judge whether a sensor node is redundant. Sensor node s_i is redundant if $S \backslash s_i$ is still a cover set for R ($S \backslash s_i$ represents the set of remaining sensors after removing s_i from S). That is, for any sensor $s_j \in S \backslash s_i$, all the elements of BVV_j in $BVT\,(S \backslash s_i, R)$ are within S_j. Furthermore, we observe that only $BVC_j(S \backslash s_i, R), j \in BVN_i(S, R)$ may be different from $BVC_j(S, R)$. Other sensors' bounded Voronoi polygons will not change after excluding sensor

s_i from S because their Voronoi neighbors don't change. We conclude the above discussion as follows.

Coverage Redundancy Condition: Given a bounded convex region R and a sensor network consisting of a set of sensors S, sensor s_i is redundant if and only if for any sensor $s_j \in BVN_i(S,R)$, all the elements of BVV_j in $BVT(S \backslash s_i, R)$ are covered by S_j.

In order to judge whether a sensor node is coverage redundant, the above condition needs to compute the Voronoi tessellation among all remaining $n-1$ sensor nodes. Therefore it will need n times computation of such Voronoi tessellation of the target region R so as to judge the redundant status of each of the n sensor nodes. The time complexity of Voronoi tessellation with n generators is about $O(nlogn)$ in the worst case. So the total time complexity is about $O\left(n^2 logn\right)$ and it is computation expensive. To reduce the computation cost, we propose the following optimization method.

Suppose we are checking whether sensor node s_i is coverage redundant. And $BVN_i(S,R)$ is s_i's Voronoi neighbor set. For any sensor node $s_j \in BVN_i(S,R)$, s_j's possible Voronoi neighbor set $BVN_j(S \backslash s_i, R)$ must be a subset of the union of $BVN_j(S,R)$ and $BVN_i(S,R)$ because of the nearest proximity characteristic of Voronoi tessellation, i.e., $BVN_j(S \backslash s_i, R) \subseteq (BVN_j(S,R) \backslash s_i) \cup BVN_i(S,R)$. This implies that $BVC_j(S \backslash s_i, R) = BVC_j(U_j, R)$, where $U_j = BVN_j(S,R) \backslash s_i \cup BVN_i(S,R)$. Therefore, it is sufficient to compute the Voronoi tessellation of R taking U_j as the generator set while determining whether s_i is coverage redundant. Using this optimization, the time complexity can be reduced to $O(nklogk)$, where $k = \max(\|U_j\|) \ll n$.

We propose the optimized coverage redundant condition as follows.

Optimized Coverage Redundancy Condition: Given a bounded convex target region R and a sensor network consisting of a set of sensors S, sensor s_i is redundant if and only if for any sensor $s_j \in BVN_i(S,R)$, all the elements of $BVV_j(U_j,R)$ are covered by s_j, where $U_j = BVN_j(S,R) \backslash s_i \cup BVN_i(S,R)$.

After all redundant sensors have been identified, we further classify redundant sensors into two sub-categories: independently and dependently redundant sensor.

Definition 4. *(Independently/Dependently redundant sensor) An independently redundant sensor is a redundant sensor whose Voronoi neighbors are all non-redundant. A dependently redundant sensor is a redundant sensor whose Voronoi neighbors include at least one redundant sensor.*

Non-redundant sensors must be kept active and independently redundant sensors are safe to be turned off. If two dependently redundant sensors, which are Voronoi neighbors each other, are turned off simultaneously, an area between them may be left uncovered, thus a sensing hole will occur. So the problem of calculating the minimal set of the active sensors is equivalent to determining the maximal set of the dependently redundant sensors that can be turned off simultaneously. In our solution, we use Redundant Dependency Graph to resolve the dependency relationship.

Definition 5. *(RDG graph) A Redundant Dependency Graph, RDG* $= (V_r, E_r)$*, is an undirected graph, where* V_r *is the set of all dependently redundant sensors and* $\forall v_{ri}, v_{rj} \in V_r, (v_{ri}, v_{rj}) \in E_r$ *if and only if* v_{ri} *and* v_{rj} *are Voronoi neighbors.*

The problem of determining the maximal set of dependently redundant sensors that can be deactivated simultaneously is equal to the problem of calculating the maximal independent set (MIS) of the RDG graph. In this paper, we adapt the greedy algorithm in [11] to calculate the MIS of the RDG graph.

When CVT algorithm terminates, all independently redundant sensors and all dependently redundant sensors selected into the MIS of the RDG graph form a Safe Set (SS), which includes all sensors that can be deactivated safely at the same time. And all other sensors form the final cover set for the target region and must be kept active to respond a specific query.

The pseudocode of the CVT algorithm is presented in Algorithm 1. The input of the CVT algorithm are the initial set of sensors $IS = \{s_1, \ldots, s_n\}$ and the target region R. Its output is a cover set (CS) for R. In the description, IRS denotes Independently Redundant sensor Set, DRS denotes Dependently Redundant sensor Set, SS denotes Safe sensor Set, S denotes temporary sensor Set and MIS denotes Maximal Independent Set.

Algorithm 1 CVT Algorithm

```
1: SS = CS = ∅, S = IS
2: if R ⊄ ⋃_{s_i ∈ S} S_i then
3:    Return CS = ∅
4: end if
5: while S ≠ ∅ do
6:    IRS = DRS = MIS = ∅
7:    Identify all redundant sensor nodes in S
8:    for each redundant sensor s_i ∈ S do
9:       if s_i is independently redundant then
10:         IRS = IRS ⋃ {s_i}
11:      else
12:         DRS = DRS ⋃ {s_i}
13:      end if
14:   end for
15:   if DRS ≠ ∅ then
16:      Compute MIS of the RDG graph
17:      SS = SS ⋃ IRS ⋃ MIS
18:      S = IS − SS
19:   else
20:      SS = SS ⋃ IRS
21:      Break out the while loop
22:   end if
23: end while
24: CS = IS − SS
```

The approximation performance of the CVT algorithm is determined by the greedy algorithm for the MIS problem. It has been shown that in any graph with N vertices, M edges and average degree δ, the greedy algorithm can take at most $O(M)$ steps to find an independent set of size at least $N/(\delta+1)$[11].

In the following, we analyze the time complexity of the CVT algorithm.

Lemma 1. *The maximum possible number of Voronoi vertices in a two dimensional Voronoi tessellation is* $2n-5$*, and the maximum possible number of Voronoi edges is* $3n-6$*, where* n *is the number of generators [10].*

Lemma 2. *The average number of Voronoi edges per Voronoi polygon does not exceed 6 [10].*

Lemma 3. *Any algorithm for constructing Voronoi tessellation requires at least* $O(nlogn)$ *time in the worst case, where* n *is the number of generators [10].*

Theorem 2. *The worst-case time complexity of the CVT algorithm is less than* $O(mnk\log k)$*, where* k *is the maximum possible number of Voronoi neighbor of each sensor,* n *is the total number of sensors initially deployed in the network,* m *is the maximum iterative times of the* ***While*** *loop of the CVT algorithm and* $m \leq \log_{7/6} n$*.*

Proof. Step $2 \sim 4$ can be completed in $O(n\log n)$. In the **While** loop, step 7 can be done in $O(nk\log k)$ using the optimized coverage redundancy condition, where k is the maximum number of Voronoi neighbors. Step $8 \sim 14$ need $O(n)$ to classify redundant sensors. Step 16 needs $O(l)$ to calculate the MIS of RDG graph, where l is the edge number of RDG and $l \leq 3n-6$. Therefore one pass of the **While** loop can be done at most in $O(nk\log k)$ in the worst case. Suppose the total iterative number of **While** loop is m. According to the approximation ratio of the greedy algorithm for the MIS problem, we can get $m \leq \log_{7/6} n$. The total time complexity of the CVT algorithm is dominated by step $5 \sim 23$ and thus is less than $O(mnk\log k)$. ■

3.2 SMT Based Connection Algorithm

It has been shown that the complete coverage of a convex region implies connectivity of the communication network induced by all active sensor nodes if $R_c \geq 2R_s$ [12]. Therefore the cover set constructed by the CVT algorithm is connected if $R_c \geq 2R_s$. But when $R_c < 2R_s$, the communication connectivity of the cover set can't be ensured. In this subsection, we propose a Steiner Minimum tree (SMT) based connection algorithm to make the cover set connected if necessary.

Definition 6. *(Primary/Assistant sensor) A sensor in the cover set is called a primary sensor. Otherwise it is an assistant sensor.*

Definition 7. *(Connected component) A connected component in the cover set is a connected subgraph of* G_c *whose vertices only involve primary sensors, where* G_c *is the communication graph induced by all initially deployed sensors.*

Suppose that sensors in the cover set constructed by the CVT algorithm form m separated connected components in G_c as shown in Fig.1. The connected component consists of only primary sensors (denoted by solid circular dots). These components are connected through assistant sensors (denoted by hollow circular dots). We abstract each connected component as one virtual vertex and use v_i to represent $comp_i$. Then we attach to v_i all edges in G_c that connect one primary sensor in $comp_i$ and another assistant sensor. After this conversion, we get an abstract communication graph $G_c{}'$ (see Fig.2).

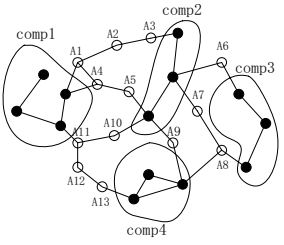


Fig. 1. Connected Components in G_c

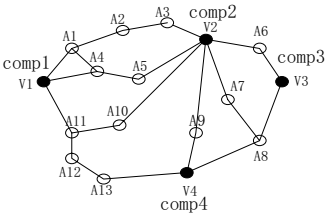


Fig. 2. Abstract Communication Graph $G_c{}'$

If we assign weight 1 to each edge in $G_c{}'$, selecting the minimal number of additional sensors needed to connect the components in G_c is equivalent to solving the Steiner Minimum Tree problem in the 1-weighted graph $G_c{}'$, where the virtual vertices representing components are considered as "terminals" and the needed assistant sensors are those "Steiner points" that are needed to construct the Steiner Minimum Tree to connect all the terminals. We present the SMT-based connection algorithm in Algorithm 2.

Algorithm 2 SMT-based Connection Algorithm

1: Create the 1-weighted abstract communication graph $G_c{}'$ as described
2: Calculate the Steiner Minimum Tree (SMT) to connect all virtual vertices in $G_c{}'$.
3: Return all the assistant sensors corresponding to the Steiner points in the computed SMT.

4 Performance Evaluation

To evaluate the performance of the proposed algorithms, we have constructed a simulator in C programming language under Linux operating system. The target region is a square of size 400×400. The default sensing radius is 30. We change the communication radius to evaluate the performance under different ratio of R_c to R_s. To ensure the target region is completely covered by the initial sensor network and the initial network is connected when all sensors are active, we

first divide the region into grid cells with dimension of 40 and deploy one sensor at the center of each grid cell. In this deployment, the minimal communication range that can ensure network connectivity is 40. When the sensing range is 30, each sensor can only cover the grid where it is located. So we can expect the minimal size of the cover set is about 100 when the sensing range is 30. Besides the 100 fixed sensors, we also randomly deploy lots of sensors in the region.

4.1 Comparison with Greedy Algorithm

The greedy algorithm in [9] is also a centralized algorithm. This algorithm firstly needs to divide the target region into many subelements, where points in the same subelement are covered by the same set of sensors.

As stated in [9], the maximum number of subelements on a 2-demensional plane with n circles is $n\,(n+1)$. When there are totally K sample points needed to be examined, the preprocessing of generating subelements takes $O\left(Kn^3\right)$ [6]. The number of sample points is related to the size of the region and the grid cell. And the size of the grid cell is closely related to the coverage accuracy. Larger cell size can result in smaller K, thus less time is needed, but the coverage performance becomes worse. In application that requires every point is covered at least by one sensor, the size of the grid cell must be small enough. On the other hand, the region monitored by sensor networks is usually vast. Therefore the number of sample points is usually several orders of magnitude of the number of the sensors in the network, i.e., $K \gg n$. Therefore the time complexity of the greedy algorithm is much larger than that of CVT algorithm.

The comparison of the size of the connected cover set constructed by the greedy algorithm and our proposed algorithms is illustrated in Fig.3. It is clear that both the CVT and CVT+SMT algorithms generate much smaller CCS than the greedy algorithm when $R_s = 30$ and $R_c = 60, 40$ respectively. While the algorithm in [9] is globally greedy, our proposed algorithm can definitely identify all non-redundant sensors that must be kept active, all independently

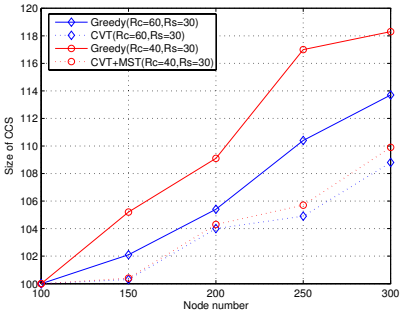


Fig. 3. Size of CCS

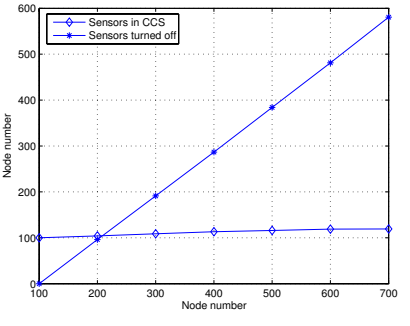


Fig. 4. Size of CCS and number of off-duty nodes vs. node density ($R_s = 30, R_c = 60$)

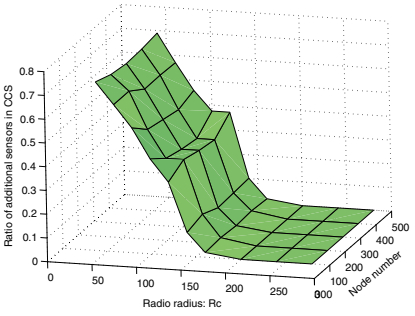


Fig. 5. Ratio of additional sensor in final CCS

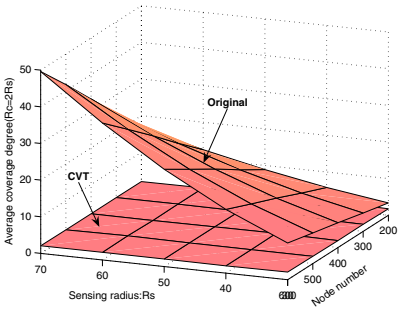


Fig. 6. Average coverage degree vs. node density

redundant sensors that can be safely turned off, and only apply the greedy algorithm to determine the maximal number of dependently redundant sensors that can be turned off simultaneously.

In summary, our proposed algorithm outperforms the greedy algorithm in terms of both the time complexity and the size of the final connected cover set (i.e., the number of sensors that must be kept active).

4.2 Simulation Results

Fig.4 shows how the size of CCS and the number of the sensors turned off vary when the number of deployed sensors changes from 100 to 700. Initially, when there are only 100 fixed sensors in the region, all sensors must be active to provide the coverage and connectivity. We observe that the size of CCS constructed by the CVT algorithm almost keeps constant and is very close to the expected number of active sensors, i.e., 100 when $R_c \geq 2R_s$. Note these 100 sensors are not the same as the 100 fixed sensor nodes.

When $R_c < 2R_s$, more additional sensors are needed to maintain the communication connectivity. Fig.5 shows the ratio of additional sensor nodes calculated by the SMT connection algorithm in the final CCS under different node density and sensing radius.

The average coverage degree can reflect the sensing redundancy of the network. To calculate the sensing degree, we divide the region into 1×1 unit cells and assume an event occurs at the center of each cell. The coverage degree is defined as the number of sensors that can detect the event. As illustrated in Fig.6, although the original sensing degree varies from 3 to 50, our proposed CVT algorithm can result in about 2 degree when $R_c = 2R_s$.

5 Conclusions

In this paper, we have designed centralized algorithms to construct a minimal connected cover set for wireless sensor networks. The basic idea is to calculate the

cover set firstly and then make it connected if needed. We propose a centralized, Voronoi tessellation (CVT) based algorithm to calculate the near optimal cover set for the target region. The constructed cover set is connected if $R_c \geq 2R_s$. In case of $R_c < 2R_s$ where the cover set constructed by the CVT algorithm is not connected, we design a Steiner minimum tree (SMT) based algorithm to compute the minimal set of additional sensors needed to make the cover set connected. Theoretical analysis and simulation results show that our proposed algorithm outperforms the greedy algorithm.

References

1. I. F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, Computer Networks (Elsevier). Vol.38 (4). March. 2002.
2. J. Elson and D. Estrin, Wireless Sensor Networks(Kluwer), 3-20 2004
3. S. Slijepcevic and M. Potkonjak. Power Efficient Organization of Wireless Sensor Networks. in Proc. of IEEE ICC'01, Helsinki, Finland, 2001.
4. F. Ye, G. Zhong, S. Lu, and L. Zhang. Peas: A Robust Energy Conserving Protocol for Long-Lived Sensor Networks. in Proc. of ICDCS, 2003.
5. D. Tian and N. D. Georganas. A Coverage-Preserving Node Scheduling Scheme for Large Wireless Sensor Networks. in Proc. of WSNA'02, Atlanta, Georgia, USA, 2002.
6. H. Chen, H. Wu, and N. Tzeng. Grid-Based Approach for Working Node Selection in Wireless Sensor Networks. in Proc. of IEEE International Conference on Communications (ICC), Paris, France, 2004.
7. B. Carbunar, A. Grama, J. Vitek and O. Carbunar, *Proceedings of the 1st IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, 377-386 (2004).
8. T. Yan, T. He, and J. Stankovic. Differentiated Surveillance Service for Sensor Networks. in Proc. of SenSys'03, Los Angels, CA, USA, 2003.
9. H. Gupta, S. R. Das, and Q. Gu. Connected Sensor Cover: Self-Organization of Sensor Networks for Efficient Query Execution. in Proc. of MobiHoc'03, Annapolis, Maryland, USA, 2003.
10. A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. Spatial Tessellations: Concepts and Applications of Voronoi Diagram. John Wiley & Sons, New York, 1999.
11. D. S. Hochbaum. Approximation Algorithms for Np-Hard Problems. PWS Publishing Company, 1995.
12. H. Zhang and J. C. Hou. Maintaining Sensing Coverage and Connectivity in Large Sensor Networks. Proceedings of NSF International Workshop on Theoretical and Algorihtmic Aspects of Sensors, Ad Hoc Wireless, and Peer-to-Peer Networks, 2004.

Near Optimal and Energy-Efficient Scheduling for Hard Real-Time Embedded Systems*

Amjad Mohsen and Richard Hofmann

Department of Communication Systems and Computer Networks,
University of Erlangen, Martensstr. 3, 91058 Erlangen, Germany
Amjad.muhsen@infromatik.uni-erlangen.de, rhofmann@cs.fau.de

Abstract. In this paper, we present a new energy-aware scheduling scheme for real-time applications using architectures that employ voltage scaling technologies. Both dynamic voltage scaling (DVS) and dynamic threshold voltage scaling (DVTHS) can benefit from this scheduling scheme. The start time of each task is adapted to enhance the efficiency of voltage scaling schemes while still satisfying the required time feasibility. The introduced scheduling scheme is able to escape local minima and it can generate near-optimal schedules in terms of energy reduction. The scheduling paradigm is integrated into our automated and multiobjective system-level co-synthesis tool that performs system optimization. We report in this paper up to about 30% higher energy reduction compared to only performance-aware scheduling.

1 Introduction

The complexity of real-time embedded systems is increasing rapidly with a non-stopping demand for higher performance. Especially for mobile systems, power consumption is a real challenge since these systems depend on a battery that can supply energy only for limited time. The steady move towards deep submicron technologies will make power rather dominate all other design constraints. Therefore, new methodologies and automated tools are required to handle design tradeoffs at all abstraction levels in order to sharply reduce the consumed energy. Without such methodologies that deal with all requirements and tradeoffs concurrently, designed systems will continue to dissipate unnecessary energy that can be saved by using more efficient and energy-aware design tools.

A prominent approach which remarkably reduces total power/energy consumption is dynamic voltage scaling (DVS). DVS was proposed in order to trade performance for power without loosing the peak performance of the system [1]. Reducing the supply voltage by a factor of two in DVS-enabled systems reduces the consumed (dynamic) energy by a factor of four. As technologies continue to scale down, the leakage power will account to about 50% of the total consumed power or even more. Applying DVS alone can achieve only linear reduction in the leakage power. An efficient scheme which can limit the leakage power is adaptive body biasing (ABB).

* We are grateful indeed that DAAD (German Academic Exchange Service) supports this research since 2002.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 234 – 244, 2005.
© IFIP International Federation for Information Processing 2005

In this scheme, the threshold voltage is modified based on demand according to the state of the system and its workload [2]. This scheme can cause exponential reduction in leakage power.

Applying voltage scaling schemes to exploit slack intervals in the system can reduce the total power/energy dissipated in digital systems. However, the availability of enough slack intervals and their distribution, which are directly influenced by the time schedule of the tasks, are crucial efficiency issues for voltage scaling schemes. Additionally, it is essential to consider power profiles of the tasks in distributed embedded systems when deriving the schedule [4]. Therefore, additional power/energy reduction can be achieved when the time schedule is planned with these issues in mind. The time schedule should be modified to enable tasks which are major consumers of energy to exploit longer slack intervals when applying voltage scaling.

This paper is organized as follows: The next section surveys some energy-aware related approaches. Section 3 reviews our energy-aware co-synthesis approach. Section 4 introduces basic concepts in voltage scaling schemes. In section 5, we present the proposed time scheduling algorithm. The obtained experimental results are presented and analyzed in section 6. We conclude this paper in section 7.

2 Related Work

Many approaches were proposed to solve the scheduling problem while considering energy issues. Gruian and Kuchcinski introduced a task scheduling heuristic based on list-scheduling [3]. The scheme dynamically calculated priorities of the tasks and tried to choose the best supply voltage levels (assuming DVS-enabled architectures) that minimize the consumed energy. The presented priority function was aware of energy aspect but might fail when the deadline was tight. To compensate for this drawback, a priority function tuning mechanism was used.

A static voltage scheduling problem was proposed and formulated as an integer linear programming (ILP) problem in [4]. The tasks were assumed to have different average switching capacitance per cycle to consider processing elements' power profiles. It was shown that considering the power profile when scaling the voltage could be a source of additional power reductions. Some other studies also reached a similar conclusion [5], [6]. However, these studies were done for single processor systems and did not show the effect of scheduling in the time domain on the performance of the voltage scheduler.

Grajcar suggested a genetic list scheduling algorithm without tackling the power problem [7]. In this approach, each individual represented an encoded schedule and the individuals of a population are ranked based on their fitness. A similar approach was used by Schmitz et al. in [8] to achieve energy–efficient scheduling when using DVS-enabled architectures. Priorities of the tasks were encoded into priority strings which are iteratively optimized by an evolutionary algorithm. The list scheduler then determines the start execution time based on the optimized priorities. The authors suggested the genetic list-based scheduling algorithm inside a global mapping optimization loop which was based on genetic algorithms.

Part of the proposed approaches may fail under stringent delay constraints. Others did not concretely tackle factors which have significant influence on total reduced

energy such as continuous slack availability and distribution. We propose in this paper a near-optimum energy-aware scheduling scheme which modifies the time schedule iteratively in order to maximize the achieved energy reduction assuming that voltage scaling enabled architectures are used. The lengths of slack intervals available in the time schedule as well as their distribution are *improved* to achieve higher energy reduction. Different from previous approaches, our scheme has the ability to escape local minima. The scheduling algorithm is integrated inside our own automated genetic-based co-synthesis tool.

3 Overview of System-Level Co-synthesis

System-level synthesis is considered here as mapping a behavioural description onto a structural specification. Functional objects have the granularity of algorithms, tasks, procedures, etc, while structural objects are processors, ASICs, buses, etc. In order to automate the synthesis process, system specification is captured into a system model.

3.1 System Model

The system is described here using two graphs: a task graph (*TG*), and an architecture graph (*AG*). Both graphs are automatically generated from SDL/MSC specification. The *TG* is a directed acyclic graph $F_p(\Psi,\Omega)$, where Ψ represents the set of vertices in the graph $(\psi_i \in \Psi)$ and Ω is the set of directed edges representing precedence constraints and data dependencies $(\omega_i \in \Omega)$. Hard real-time constraint(s) T_i is/are forced on a node or a set of nodes as well as an absolute time constraint T_T. The *AG* is $F_A(\Theta,\Re)$, where Θ stands for the set of available architectures $(\theta_i \in \Theta)$ and $(\rho_i \in \Re)$ represents possible connections between hardware components. For each component $(\theta_i \in \Theta)$, a finite set of resource types (S) is defined. For each resource type ($s_i \in S$) there is a set of associated ratios (R_s) that specify power-, delay-, and cost-scaling when using this type.

3.2 Optimization Approach

System-level synthesis requires optimizing three basic sub-problems: First, allocation (α) which selects a set of hardware components for implementation purposes. Second, mapping (β) which determines which single hardware resource is used to execute each task. Based on the allocation and mapping, the schedule of tasks is then optimized to achieve time feasibility and to maximize energy saving when applying voltage scaling. In our approach, a voltage schedule specifies the voltage levels required to execute each task. Therefore, the schedule (sc) includes a time and a voltage schedule. The time schedule is optimized to achieve time feasibility and to maximize simultaneously energy reduction when applying voltage scaling.

The synthesis process is considered in our approach as a multiobjective optimization problem which can be stated as: *minimize the function* $f(\alpha,\beta,sc) = (f_1(\alpha,\beta,sc), f_2(\alpha,\beta,sc), f_3(\alpha,\beta,sc))$ while all design constraints are satisfied. This formulation considers three objectives: performance (f_1), power (f_2), and cost (f_3). The proposed synthesis methodology applies evolutionary algorithms (EAs) to optimize

the allocation/binding (global optimization) based on the objective vectors (f_1, f_2, f_3). The schedule is optimized for each individual separately as an integrated part of the global optimization loop. EAs are suitable for such optimization problems since they work in parallel on a population of individuals. Each individual encodes a potential implementation. Also, EAs work well on problems with large search spaces.

The EA consists of an optimization loop inside which the principles of reproduction, crossover and mutation are applied on the population's individuals. The purpose is to find iteratively better population based on a set of design objectives. Each individual is evaluated and assigned a fitness value based on the objective vector and fittest individuals are allowed to mate. Fitness values reflect the superiority of each individual in terms of *all* objectives. Computing fitness and performing selection are carried out by the widely used evolutionary optimizer, SPEA2 which is integrated to our synthesis tool. SPEA2 is an up-to-date optimizer which offers several merits over normal evolutionary approaches [9]. The optimization loop terminates when the pre-specified condition(s) is/are satisfied. A set of implementation alternatives which spans the Pareto-optimal front are produced. Each implementation alternative includes an allocation, a mapping, and a time- and voltage-schedule. The designer selects one of these design alternatives based on market requirements.

4 Voltage Scaling

Because of its quadratic relationship to power, voltage reduction has the most drastic means of reducing total consumed power/energy. DVS increases the energy efficiency of a device by dynamically adapting its speed and voltage as needed. However, in heterogeneous systems, the variation in the power profiles of the tasks when executed on different components influences the amount of energy reduction that can be achieved when scaling the voltage. This can be seen in the equation below:

$$E'(\psi_i) = \left(\frac{V_{level}^2}{V_{supply}^2} \right) E(\psi_i) \Big|_{V=V_{supply}} \tag{1}$$

In the above equation, $E'(\psi_i)$ refers to the energy consumed by task ψ_i after scaling the operating voltage to V_{level} ($V_{level} \leq V_{supply}$). $E(\psi_i)$ stands for the energy consumed by this task at nominal voltage (V_{supply}) based on a certain mapping. Therefore, the higher the consumed power and the lower the adopted voltage level to execute a task the more energy reduction is achieved.

Voltage scaling schemes can also be applied to limit the amount of consumed leakage power in digital circuits. Dynamic threshold voltage (V_{th}) scaling (DVTHS) is a candidate scheme which dynamically adapts V_{th} by means of varying the body bias voltage (V_{bs}) in order to save leakage power. The relationship between V_{bs} and the consumed static power (P_{static}) is given below:

$$P_{static} = V_{dd} K_3 e^{K_4 V_{dd}} e^{K_5 V_{bs}} + |V_{bs}| I_j \tag{2}$$

In this equation, V_{dd} is the supply voltage, K_3, K_4, K_5, are constant fitting parameters which are technology dependent, and I_j is the current due to junction leakage [2].

Voltage scaling is an efficient method to reduce the total consumed power/energy. Nevertheless, scaling the voltage has negative influence on circuit delay which can be formulated as follows [2]:

$$D = \frac{K_6 L_d V_{dd}}{((1+K_1)V_{dd} + K_2 V_{bs} V_{bs} - V_{th1})^{\alpha}} \tag{3}$$

In this equation, K_1, K_2, K_6, and V_{th1} are circuit dependent constants. L_d is the logic depth and α is the logic depth which is technology dependent ($1.4 \leq \alpha \leq 2$). Hence, voltage scaling mechanisms can only be applied if degradation in performance can be tolerated. Key efficiency factors for voltage scaling schemes are the availability of enough continuous slack intervals and the way in which these intervals are exploited. The proposed scheduling scheme which is presented in the next section handles these issues efficiently. It constructs a time schedule which considers future needs of voltage scaling schemes. Without any loss of generality, we examined the proposed algorithms by applying DVS only.

5 Energy-Efficient Time Scheduling

Scheduling in our case can be defined as determining the start time and the required voltage level(s) needed to execute each task. Precedence constraints and data dependencies must be fulfilled. Hence, the problem can be seen here as a two dimensional (2-D) optimization problem. The start time of each task ψ_i, $\tau_i(t)$, and the voltage level(s) $V(\tau) = \{v_1, v_2, \ldots, v_n\}$ should be optimized to maximize the power/energy reduction. In our approach, the time schedule is determined first with an *eye on energy*. Based on the derived time schedule, the voltage schedule is determined based on a global view of all tasks. So, the objective is to optimize the start execution time of each task such that more energy reduction can be achieved when applying voltage scaling schemes. In this section, the time scheduling is presented whereas the proposed voltage scheduling algorithm can be found in [10].

As mentioned previously, the available slack intervals and their distribution among the tasks are basic efficiency factors for voltage scaling schemes. Considering power profiles of different processing elements when executing different tasks makes the influence of slack distribution more crucial. Therefore, the time schedule has to be adapted such that tasks which are major consumers of energy exploit longer slack intervals. Moreover, tasks which consume more energy should have higher priority to scale their operating voltage when deriving the voltage schedule. The proposed scheduling algorithm considers all these issues as explained below.

5.1 Scheduling Algorithm

The proposed time scheduling algorithm considers *future* needs of voltage scaling schemes in order to maximize energy reduction. The algorithm has an iterative nature and it is based on the paradigm originally proposed by Kernighan and Lin (KL) for graph partitioning problem (min-cut partitioning) [11]. Based on ASAP and ALAP

schedules, and the mobility value of each task, initial priorities are computed for all tasks. A list-based scheduler is then used to generate an initial feasible schedule. According to ASAP and ALAP, a task can be scheduled earlier or later as long as no data dependency or delay constraint will be violated. For example, task ψ_8 can be scheduled between 12 and 22 in Fig. 3.

```
Input: F_P(Ψ,Ω), F_A(Θ, Ϡ), Mapping
Output: Time Schedule: UpToDate_sched

REPEAT
   Unlock all tasks;
   Work_sched = UpToDate_sched;
   C = 0;
   WHILE There is a movable task
      C = C + 1;
       Power_gain[C]  = - ∞ ;
      FOR each task(i) ∈ {movable tasks}
         FOR each scheduling step in the ready list (step_j,r)
            Temp_sched = MOVE_TASK(Work_sched, task(i), step_j,r);
            Temp_gain = POWER(Work_sched) – POWER(Temp_sched);
            IF Temp_gain > Power_gain[C];
                 Power_gain [C] =  Temp_gain;
                 Sequence [C] = task(i);
                 Sched_step[C] = step_j,r;
            ENDIF
         ENDFOR
      ENDFOR
      Lock (moved tasks)
      Work_sched=MOVE_TASK(Work_sched, Sequence [C], Sched_step[C]);
      ENDWHILE
      Select a sequence q based on Power_gain[];
      IF maximum cumulative power reduction (Po_R)> 0
         Update the UpToDate_sched with the sequence;
      ENDIF
UNTIL Po_R < 0;
```

Fig. 1. Energy-aware time scheduling algorithm

The proposed scheduling algorithm iteratively selects a task and a scheduling step based on the ready list of an allocated hardware such that no delay constraint will be violated. The scheduling step that maximizes the energy reduction (when applying voltage scaling) is fixed for this task and the task itself is locked and disallowed to move later unless all tasks are locked. One task is moved at a time and each move leads to a new schedule. The new (*temporal*) schedule is repeatedly evaluated after scaling the voltage. After a set of m such moves, a subsequence of $q \leq m$ that maximizes the total energy reduction can be reached. The schedule is then changed to

include these q moves. The process is repeated and all tasks are unlocked until no further improvement in terms of energy could be achieved. This scheduling approach has been chosen because KL is known to find good solutions in small CPU time and it has been successfully applied for long time in various placement and routing applications. At the same time, this methodology allows *bad* solutions to arise and then chooses the set of moves that leads to the maximum energy reduction.

The entire scheduling algorithm is shown in details in Fig. 1. In this algorithm, Work_sched and Temp_sched are temporary schedules whereas UpToDate_sched represents the maintained up-to-date schedule. Power_gain[] stores the power gain for each iteration. The function MOVE_TASK(Work_sched, task(i), step$_{j,r}$) returns a new schedule after modifying Work_sched by scheduling task(i) into step$_{j,r}$; j represents a scheduling step and r stands for the selected ready list. The *function* POWER() takes a schedule and computes its power/energy consumption. The two arrays Sequence[] and Sched_step[] maintain the sequence of moved tasks and the corresponding scheduling steps, respectively.

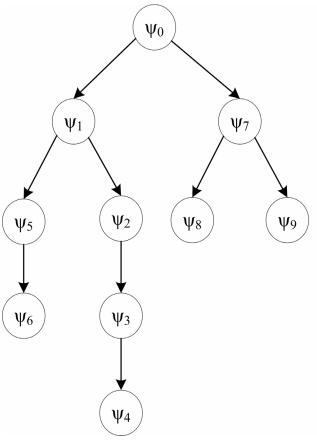
The iterative nature of the algorithm might appear to be a disadvantage. When all possible moves for each task are considered the time complexity will be $O(n)$, where n is the total number of tasks and moves. At the same time, implementing DVS requires substantial algorithms to determine the required operating voltage and the corresponding speed/frequency. These algorithms affect the performance of the device and increase the overall consumed energy. In our approach, we emphasize that scheduling processes are off-line processes that optimize start time and voltage level(s) required to execute each task during the design phase. The planned voltage levels and the time schedule are stored in a table form for run-time use. As a result, the additional energy consumption related to the scheduling process is omitted and the run-time performance overhead due to this additional process is reduced to a minimum. However, voltage switching cost in terms of energy is still considered during run-time.

5.2 Illustration Example

To demonstrate the influence of the proposed energy-ware time schedule consider the task graph presented in Fig. 2. Tasks' mapping and other supporting information regarding worst case execution time (WCET) and power consumption are found in the table shown on the right hand side in the same figure. Fig. 3 shows an initial time schedule based on the information presented in Fig. 2. The time schedule is derived using a list-based scheduling algorithm and it is presented here as a Gantt chart.

Columns 3 and 4 in the table shown in Fig. 1 give the WCET and the average power consumption for each task according to the given mapping, respectively. The start time for each task according to "as soon as possible" (ASAP) and "as late as possible" (ALAP) scheduling algorithms are given in columns 5 and 6, respectively. The mobility which is defined in column 7 of the same table is defined as the difference between ALAP and ASAP. The priority of each task is calculated based on the mobility. Based on the table shown in the above figure, it can be seen that tasks which have major energy reduction effects (high power profiles), such as tasks ψ_6, ψ_8, and ψ_9, have low priority. This means that they are scheduled in time after other

higher priority tasks. As it can be seen in Fig. 3, task ψ_6 has no slack at all while tasks ψ_8 and ψ_9 have both 6 slack slots assuming that the deadline is 26. No other task can make any benefit from voltage scaling schemes.



Task	Mapping	WCET	Power	ASAP	ALAP	Mobility
ψ_0	HW1	2	10	0	9	9
ψ_1	HW1	5	15	2	11	9
ψ_2	HW2	4	10	7	16	9
ψ_3	HW2	2	5	11	20	9
ψ_4	HW2	4	10	13	22	9
ψ_5	HW2	4	10	7	17	10
ψ_6	HW1	5	25	11	21	10
ψ_7	HW1	5	15	2	17	15
ψ_8	HW0	4	50	7	22	15
ψ_9	HW0	4	50	7	22	15

Fig. 2. Task graph and mapping example

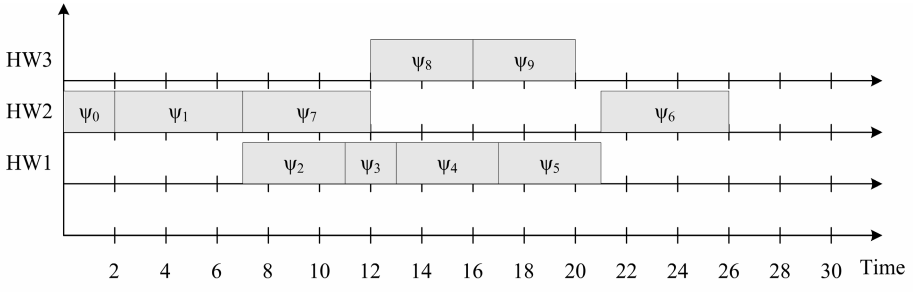


Fig. 3. Initial schedule

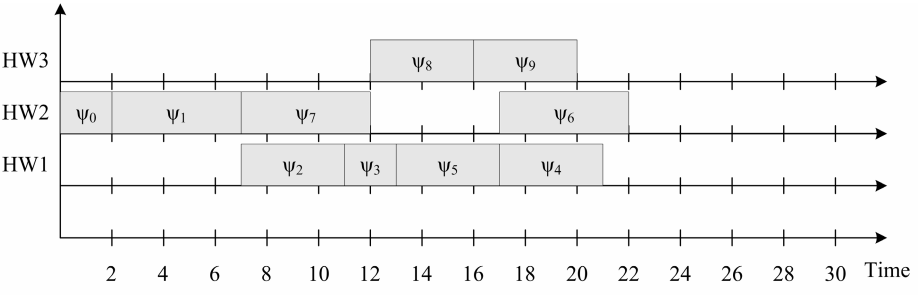


Fig. 4. Modified schedule

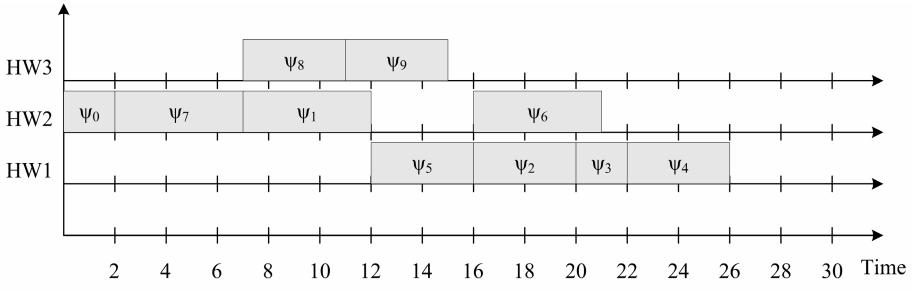


Fig. 5. An energy-efficient time schedule

Consider now the time schedule in Fig. 4 where task ψ_5 is scheduled before task ψ_4, independent on their priorities. As it can be seen in the figure, additional 4 slack slots are now available for task ψ_6 which causes higher energy reduction when applying voltage scaling. A more efficient time schedule from the energy point of view is shown in Fig. 5. In this figure, 9 time slots are now available to tasks ψ_8 and ψ_9. Task ψ_6 has now 5 time slots which can be exploited when applying voltage scaling. In this schedule ψ_7, for example, is scheduled before ψ_1 although the later has higher priority.

6 Experimental Results

A set of benchmarks are used to prove the applicability and the efficiency of our approach in reducing the total consumed energy. The benchmarks include a set of 20 publicly available benchmarks which are generated originally by using "Task Graphs For Free" TGFF [12] and obtained from [8].

Our 2-D scheduling methodology has been integrated to our automated co-synthesis tool which is presented previously. For comparison purposes, another energy-efficient time scheduling algorithm was also implemented and integrated to our tool. This later time scheduling algorithm is called evolutionary list-based scheduling algorithm (ELSA). This algorithm is also based on a list scheduling algorithm. Priorities of tasks are optimized using an up-to-date evolutionary algorithm. An optimization objective is increasing the energy reduction when applying voltage scaling. Based on the *optimized* priorities, the list schedule determines the start execution time for each task. An energy-ware scheduling algorithm similar to ELSA was proposed in [8].

In order to enable fair comparison, the 2-D and ELSA schemes were tested under the same conditions. To achieve this, we implemented ELSA and integrated it to our global optimization tool. At the same time, the same voltage scheduling algorithm was applied in all experiments. To further demonstrate the influence of energy-aware scheduling, another separate experiment was conducted. In this experiment, a traditional list scheduling algorithm was implemented to optimize the time scheduling without considering energy. We call this scheme the 1-D scheduling. Results of this experiment are shown in columns 2 and 6 of Table 1. Columns 3 and 7 present the

energy reduction achieved when applying ELSA whereas energy reductions obtained when applying the 2-D scheme are shown in columns 4 and 8 for all the included benchmarks. Obtained energy reductions are reported here in percentage.

Table 1. Energy reduction in % obtained when using 1-D, ELSA, and 2-D scheduling

Benchmark	1-D	ELSA	2-D	Benchmark	1-D	ELSA	2-D
tgff1	68.1	82.6	83.5	tgff11	24.3	30.1	33.6
tgff2	36.4	43.9	49.1	tgff12	62.6	74.6	76.0
tgff3	64.6	70.8	75.9	tgff13	60.9	72.0	73.2
tgff4	82.6	88.0	90.1	tgff14	10.0	26.4	27.6
tgff5	60.1	61.1	61.1	tgff15	14.1	27.4	27.9
tgff6	83.5	87.8	90.1	tgff16	27.4	37.9	39.1
tgff7	30.2	43.3	45.0	tgff17	40.0	58.8	63.1
tgff8	76.6	76.7	77.0	tgff18	31.0	43.5	43.6
tgff9	37.3	38.1	45.0	tgff19	47.0	58.3	76.7
tgff10	19.6	31.7	33.7	tgff20	78.5	84.1	86.6

The above presented results show that all benchmarks made benefit of the 2-D scheduling scheme, but in varying degrees. The maximum energy reduction (90.1%) is obtained for tgff4 and tgff6. The presented results indicate that the 2-D scheduling scheme can perform at least as good as the ELSA algorithm. Additional energy reduction of up to 18.4% could be achieved by the 2-D scheduling over ELSA. This can be related to fact that the 2-D scheduling scheme can escape local minima. It accepts unacceptable moves for the tasks as long as these belong to a scheduling sequence that maximizes the overall energy reduction. The results obtained when applying the 2-D scheme shows that up to about 30% higher energy reduction can be achieved over 1-D scheduling (TGFF19). ELSA can only achieve up to about 19% higher energy reduction compared to the 1-D scheduling scheme (tgff17).

7 Summary and Conclusion

This paper presented a new two-dimensional (2-D) scheduling scheme that is able to remarkably increase energy reduction obtained when applying voltage scaling. This scheduling scheme has the tendency to reach near-optimal schedules in terms of energy when applying voltage scaling. This is achieved by adopting the set of task-moves that lead to a maximum *cumulative* energy reduction which enables the algorithm to escape local minima. Experimental results indicate that up to about 30% energy reduction could be achieved.

A basic conclusion that can be drawn based on the experimental results is that it is essential to consider energy-related issues when optimizing the time schedule. Time

feasibility should firstly be guaranteed but at the same time the schedule should be adapted based on certain optimization criterion to increase its energy-efficiency.

References

1. Pering, T., Burd, T., Broderson, R.: Dynamic Voltage Scaling and the Design of a Low-Power Microprocessor System. In *Power Driven Micro-Architectures Workshop*, attached to ISCA'98, Barcelona, Spain, June, (1998).
2. Martin, S., Flautner, K., Mudge, T., Blaauw, D.: Combined Dynamic Voltage Scaling and Adaptive Body Biasing for Lower Power Microprocessors under Dynamic Workloads. In *Proceedings of the International Conference on Computer-Aided Design***,** ICCAD'02, (2002) 721-725.
3. Gruian, F., Kuchcinski, K., LEnS: Task Scheduling for Low-Energy Systems Using Variable Supply Voltage Processors. In *Proceedings of Asia and South Pacific Design Automation Conference*, ASP-DAC, January (2001) 449-455.
4. Ishihara T., Yasuura, H.: Voltage Scheduling Problem for Dynamically Variable Voltage Processors. In *Proceedings of the International Symposium on Low Power Electronics and Design*, ISLPED, (1998) 197-202.
5. Okuma, T., Ishihara, T., Yasuura, H.: Real-Time Task Scheduling for a Variable Voltage Processor. In *Proceedings of the 12th International Symposium on System Synthesis***,** ISSS, (1999) 24-29.
6. Manzak A., Chakrabarti, C.: Variable Voltage Task Scheduling for Minimizing Energy or Minimizing Power. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, November (2000) 3239-3242.
7. Grajcar, M.: Genetic List Scheduling Algorithm for Scheduling and Allocation on a Loosely Coupled Heterogeneous Multiprocessor System. In *Proceedings of the 36th ACM/IEEE Conference on Design Automation*, (1999) 280-285.
8. Schmitz, M., Al-hashimi, B., Eles, P.: Energy-Efficient Mapping and Scheduling for DVS Enabled Distributed Embedded Systems. In *Proceedings of Design, Automation and Test in Europe Conference and Exhibition, DATE,* March, (2002) 514-521.
9. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In *Proceedings of Evolutionary Methods for Design, Optimization, and Control, CIMNE,* Barcelona, Spain, (2002) 95-100.
10. Mohsen, A., Hofmann R.: Efficient Voltage Scheduling and Energy-aware Co-synthesis for Real-time Embedded Systems. Tenth Asia-Pacific Computer Systems Architecture Conference, Singapore, October 24-26, 2005.
11. Kernighan K., Lin, S.: An Efficient Heuristic Procedure for Partitioning Graph. *Bell System Technical Journal*, vol. 49, no. 2, (1970) 291-307.
12. Dick, R., Rhodes, D., Wolf, W.: TGFF: Tasks Graphs for Free. In *Proceedings of International Workshop on Hardware/Software Codesign*, March, (1998).

Performance Evaluation of Power-Aware Communication Network Devices

Hiroyuki Okamura and Tadashi Dohi

Department of Information Engineering,
Graduate School of Engineering,
Hiroshima University, Higashi-Hiroshima, 739–8527, Japan
{okamu, dohi}@rel.hiroshima-u.ac.jp

Abstract. In this paper, we focus on power-aware network devices, and propose a stochastic model to evaluate their performance quantitatively. More precisely, applying the Markovian additive process (MAP) to the arrival stream for network devices, we develop a stochastic dynamic power management (DPM) model with shutdown policy. Two performance measures for power-saving and processing; steady-state power consumption and throughput, are derived analytically for the DPM model. In numerical examples, we investigate the performance of the power-aware communication device with shutdown in terms of power-saving and processing.

1 Introduction

Energy consumption is a critical issue for ubiquitous computing such that a huge number of computer systems are embedded everywhere. Since most ubiquitous systems have to process the tasks imposed with the limited amount of power supplied by battery packs, both technologies on power reduction and power saving are important to design the ubiquitous systems with embedded, portable and wearable architecture. Thus the power management to reduce energy consumption has received considerable attention for recent years.

Generally speaking, since computer systems, including ubiquitous systems, consist of a number of electric components and devices, the main issues on power management are to reduce energy consumption in each component such as IC chip, microprocessor, CPU, disk drive, display *etc.* To save energy consumption for a lot of devices, the software-based power management technique seems to be more effective than the hardware-based power reduction from the view points of cost and flexibility. In fact, note that the devices installed in most systems are capable to reconfiguration by software control. The simplest but effective idea for saving electricity is to stop the electricity supply for the devices when they become idle. In other words, when the devices terminate their own tasks, the controller of software-based power management, *i.e.* a power manager (PM), should shut down the devices immediately. Such a software-based power management is often called *dynamic power management* (DPM). DPM is a control scheme that reconfigures components dynamically to serve the tasks imposed

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 245–254, 2005.
© IFIP International Federation for Information Processing 2005

with a minimum number of active components [1, 3]. In DPM, two approaches on reduction of power consumption are available: *shutdown* and *variable-voltage*. The shutdown approach is the simplest way for the power reduction, and is to shut down a device when it has been idle for long time. On the other hand, the variable-voltage approach is to reduce the supply voltage for processors. That is, adjusting system speed at the minimum level necessary to processing the tasks leads to drastically reducing energy consumption.

Benini *et al.* [1] discuss the system-level DPM. In their modeling, the system can be mainly classified into three states: active, idle and sleep. In this framework, by taking account of trade-off relationship between processing performance and energy consumption, PM can control the system under the DPM. For instance, when applying the shutdown and the variable-voltage, we can carry out the performance evaluation under these basic DPM models. In particular, Benini *et al.* [3] describe that the control with fixed shutdown timing takes an advantage of the implementation, but in the fixed shutdown policy, the inadequate (shorter or longer) timing of shutdown, may cause a failure for reducing energy consumption. That is, it implies that there is an optimal control policy to save energy consumption effectively in DPM. Srivastava *et al.* [12] propose predictive shutdown policies to improve the fixed shutdown timing by making decisions as soon as the device becomes idle, based on the successive observations for active and idle states. The shutdown policy by Hwang and Wu [5] uses the weighted sum of the last idle period and the last prediction as the upcoming prediction of the idle period, and improves ones in [12]. Unlike these models, the stochastic DPM models are considered in [2, 4, 10]. Qiu and Pedram [10] and Benini *et al.* [2] study DPM models with shutdown by applying the Markov decision processes. In their models, the workload is modeled by a simple two-state Markov chain, and the problem is formulated as choosing the optimal strategies on each state. Chung *et al.* [4] point out that the policy optimization in [2, 10] is based on a priori knowledge of the system and its workload statistics, but that the workload is generally much harder to characterize in advance. Since workloads should be often non-stationary in general, they consider a DPM model for non-stationary service requests. Similar shutdown approaches can be applied to the hard disk power management on personal computers. Sandoh *et al.* [11] develop a simple renewal reward model and derive the optimal timing at which the hard disk device should go to the sleep state. Lu and De Micheli [6] and Lu *et al.* [7] present the computation algorithms for controlling the power states of a hard disk device. On the other hand, as the DPM models based on stochastic processes, Okamura *et al.* [8, 9] develop some shutdown models based on the renewal structure and queueing theory to represent the number of arrival requests, *i.e.* arrival stream of requests, more strictly. In their works, under the circumstance that the arrival process of requests is according to the Poisson process, it is optimal to shut down the device just after each task is terminated, or not to shut the device at all, depending on the arrival rate.

This paper focuses on power-aware communication network devices such as a wireless LAN. In such devices, the problem on power reduction is critical, because

the energy consumption of network devices cannot be ignored for the total consuming electricity for mobile system. After developing a stochastic DPM model for the network device and the mobile communication environment, we derive analytically some performance measures on both power-saving and processing. As mentioned before, if the arrival stream is given by the Poisson process, it is proved that the optimal shutdown timing becomes zero or infinity. However, in the practical communication network, the arrival stream for devices might not obey a simple Poisson process, because there is the correlation of requests. This implies that there are the states of high and low arrival rates in the communication network. Thus it is worth noting to consider the modeling with the correlation of requests. For the modeling with correlation, the Markov modulated Poisson process (MMPP) is often used. The MMPP is the stochastic process in which the arrival rate changes according to a continuous-time Markov chain (CTMC), and is one of the simplest stochastic processes to represent correlation of accesses. Since the Markovian additive process (MAP) includes the MMPP as a special case and allows the arrival rate to change at the occurrence of an arrival, it can model the more complex correlation of accesses.

2 Model Description

2.1 Modeling of Arrival Stream

First we model an arrival stream of requests in the underlying network device. Suppose that an arrival stream for the underlying device has a number of internal states which are called phases, and that the transition of phases occurs according to a CTMC. Let $\{J(t); t \geq 0\}$ be the CTMC which indicates the phase at time t, and the corresponding infinitesimal generator is given by

$$\boldsymbol{M} = \begin{pmatrix} -\sum_j \mu_{1,j} & \mu_{1,2} & \cdots & \mu_{1,m} \\ \mu_{2,1} & -\sum_j \mu_{2,j} & \cdots & \mu_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ \mu_{m,1} & \mu_{m,2} & \cdots & -\sum_j \mu_{m,j} \end{pmatrix}. \tag{1}$$

For sake of simplification, the number of phases, m (≥ 1), is to be finite. The arrival rate of requests depends on the phase, and is governed by a CTMC of phases. That is, from the Markovian property, the arrival rate can be determined only by the current phase. Define the following matrix whose element indicates the arrival rate of request on the corresponding phase:

$$\boldsymbol{D} = \begin{pmatrix} \lambda_{1,1} & \lambda_{1,2} & \cdots & \lambda_{1,m} \\ \lambda_{2,1} & \lambda_{2,2} & \cdots & \lambda_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{m,1} & \lambda_{m,2} & \cdots & \lambda_{m,m} \end{pmatrix}. \tag{2}$$

The diagonal elements of the matrix $\boldsymbol{D}$ indicate the occurrence rates of requests corresponding to each phase. This rate does not include the occurrence rate of

phase transition, *i.e.* the corresponding event is that one request occurs but the phase does not change. On the other hand, non-diagonal elements represent the rates of the simultaneous events that a request occurs and the phase changes.

Let $\{N(t); t \geq 0\}$ denote the cumulative number of requests before time t and $\boldsymbol{P}(t)$ be the matrix whose (i,j)-element is given by

$$p_{i,j}(t) = \Pr\{N(t) = n, J(t) = j \mid N(0) = 0, J(0) = i\} \quad i,j = 1,2,\ldots,m. \tag{3}$$

From the matrix-based Markovian analysis, we obtain the following differential-difference equations:

$$\frac{d}{dt}\boldsymbol{P}_0(t) = \boldsymbol{P}_0(t)\boldsymbol{C}, \tag{4}$$

$$\frac{d}{dt}\boldsymbol{P}_k(t) = \boldsymbol{P}_k(t)\boldsymbol{C} + \boldsymbol{P}_{k-1}(t)\boldsymbol{D}, \quad k = 1,2,\ldots \tag{5}$$

and

$$\boldsymbol{C} = \boldsymbol{M} - \mathrm{diag}(\boldsymbol{D}\boldsymbol{e}), \tag{6}$$

where $\boldsymbol{e}$ is the column vector whose elements are 1 and $\mathrm{diag}(\boldsymbol{a})$ gives a matrix with the elements of $\boldsymbol{a}$ on the leading diagonal. Then the stochastic process $N(t)$ is called the Markovian additive process (MAP) with parameters $\boldsymbol{C}$ and $\boldsymbol{D}$.

For example, letting $\boldsymbol{C} = -\lambda$ and $\boldsymbol{D} = \lambda$, the corresponding MAP is reduced to the homogeneous Poisson process with the arrival rate λ. Also, we can represent the switched Poisson process (SPP), interrupted Poisson process (IPP) and the Markov modulated Poisson process (MMPP) with the same framework. Further the MAP models an arrival stream in the practical situation. Actually, if we set the parameters as,

$$\boldsymbol{C} = \begin{pmatrix} 0 & 0 \\ \mu & -(\lambda_a + \mu) \end{pmatrix}, \quad \boldsymbol{D} = \begin{pmatrix} 0 & \lambda_s \\ 0 & \lambda_a \end{pmatrix}, \tag{7}$$

then the associated MAP can represent the situation where a communicative session starts and terminates at the rates λ_s and μ, respectively, and the requests arrive at the underlying network device with the rate λ_a while the session is alive.

An advantage of the MAP is to represent convolutional of arrival streams easily. Let $\boldsymbol{C}_k$ and $\boldsymbol{D}_k$, $k = 1,\ldots,K$ denote the parameters of MAPs representing the arrival streams by respective peripheral devices. Then the total number of requests for the underlying devices is given by the convolutional of these MAPs with the parameters

$$\boldsymbol{C} = \boldsymbol{C}_1 \oplus \cdots \oplus \boldsymbol{C}_K, \quad \boldsymbol{D} = \boldsymbol{D}_1 \oplus \cdots \oplus \boldsymbol{D}_K, \tag{8}$$

where $\oplus$ is the Kronecker sum.

2.2 Modeling of the Network Device

Consider a network device which has the three energy consumption states: active, idle and inactive (sleep). The total number of requests from other devices follows

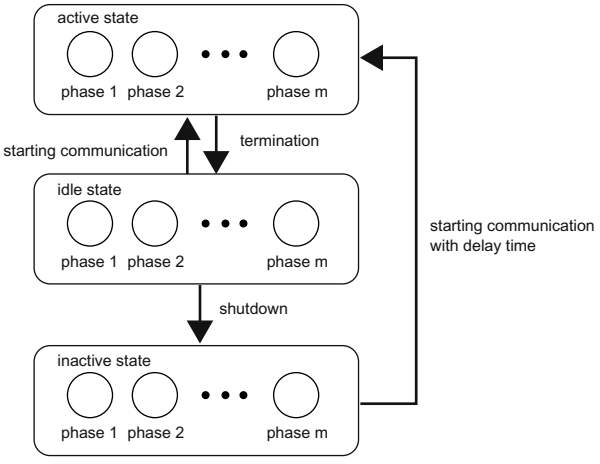


Fig. 1. Configuration of state transition in DPM model

the MAP with parameters $\boldsymbol{M}$ and $\boldsymbol{D}$. To simplify the discussion, the arrival stream has m (> 0) different states of phases. When a request arrives at the underlying network device, the device communicates with the corresponding device. The time required to communicate the other is given by a random variable S having the probability distribution function $F(t)$. During the communication, the other succeeding requests are completely refused. After the communication is terminated, the device waits for next request in the idle state. If no access occurs for a limited time period t_0, PM shuts down the device to save the electricity, *i.e.* the state of device becomes inactive. In the inactive state, when a request occurs, the device starts to communicate with the other device, where it takes a random time delay D to set up for the communication, having the probability distribution function $H(t)$. On energy consumption, it is assumed that the amounts of energy consumption per unit time in active, idle and inactive states are given by P_1 (> 0), P_2 ($0 < P_2 < P_1$) and 0, respectively. Figure 1 shows the state transition diagram of DPM model.

3 Performance Measures

Here two kinds of performance measures will be considered in terms of power saving and processing: (i) Steady-state power consumption and (ii) Throughput. To derive these performance measures, we apply the hidden Markovian method. Define the time period between two successive terminations of communication as one cycle. Taking account of the state transition of phase at each termination, the temporal behavior of phase transition is described by a discrete-time Markov chain (DTMC). Let $\boldsymbol{G}_0(t_0)$ denote the transition probability matrix for phases at time instance of communicative termination, provided that the shutdown time

is given by t_0. From the matrix-based Markovian analysis [13], the transition probability matrix is given by

$$\begin{aligned}\boldsymbol{G}_0(t_0) &= \int_0^{t_0} \exp(\boldsymbol{C}t)dt\boldsymbol{D}\boldsymbol{F} + \int_{t_0}^{\infty} \exp(\boldsymbol{C}t)dt\boldsymbol{D}\boldsymbol{H}\boldsymbol{F} \\ &= \{\boldsymbol{I} - \exp(\boldsymbol{C}t_0)\}(-\boldsymbol{C})^{-1}\boldsymbol{D}\boldsymbol{F} + \exp(\boldsymbol{C}t_0)(-\boldsymbol{C})^{-1}\boldsymbol{D}\boldsymbol{H}\boldsymbol{F} \\ &= (-\boldsymbol{C})^{-1}\boldsymbol{D}\boldsymbol{F} - \exp(\boldsymbol{C}t_0)(-\boldsymbol{C})^{-1}\boldsymbol{D}\{\boldsymbol{I} - \boldsymbol{H}\}\boldsymbol{F}. \end{aligned} \tag{9}$$

In this equation, the matrices $\boldsymbol{H}$ and $\boldsymbol{F}$ denote the transition probability matrices for the delay time and the communication time, respectively, where

$$\boldsymbol{H} = \int_0^{\infty} \exp(\boldsymbol{Q}t)dH(t), \quad \boldsymbol{F} = \int_0^{\infty} \exp(\boldsymbol{Q}t)dF(t), \tag{10}$$

and $\boldsymbol{Q} = \boldsymbol{C} + \boldsymbol{D}$. Let us define the steady-state probabilities for phases on the hidden Markov chain, provided that the shutdown timing is t_0 as

$$\boldsymbol{\pi}(t_0) = [\pi_1(t_0), \pi_2(t_0), \ldots, \pi_m(t_0)]. \tag{11}$$

Then it is evident that

$$\boldsymbol{\pi}(t_0) = \boldsymbol{\pi}(t_0)\boldsymbol{G}_0(t_0), \quad \boldsymbol{\pi}(t_0)\boldsymbol{e} = 1. \tag{12}$$

In fact, the computation in Eq. (12) can be reduced into the problem on eigenvalue and eigenvectors for matrices.

(i) Steady-state power consumption

First, we consider the expected time length of one cycle. Since the expected time length depends on the state of phase at the beginning of cycle, *i.e.*, the termination of communication, the expected time length of one cycle is represented by a column vector. When the shutdown timing is given by t_0, the expected time length of one cycle becomes

$$\begin{aligned}\boldsymbol{\tau}(t_0) &= \int_0^{t_0} \{t + \mathrm{E}[S]\} \exp(\boldsymbol{C}t)\boldsymbol{D}\boldsymbol{e}dt \\ &\quad + \int_{t_0}^{\infty} \{t + \mathrm{E}[D] + \mathrm{E}[S]\} \exp(\boldsymbol{C}t)\boldsymbol{D}\boldsymbol{e}dt \\ &= (-\boldsymbol{C})^{-2}\boldsymbol{D}\boldsymbol{e} + \mathrm{E}[S]\boldsymbol{D}\boldsymbol{e} + \mathrm{E}[D] \exp(\boldsymbol{C}t_0)(-\boldsymbol{C})^{-1}\boldsymbol{D}\boldsymbol{e}. \end{aligned} \tag{13}$$

Similarly, the expected power consumption during one cycle also depends on the state of phases at the beginning of cycle, and is given by

$$\begin{aligned}\boldsymbol{\xi}(t_0) &= \int_0^{t_0} \{P_1 t + P_2 \mathrm{E}[S]\} \exp(\boldsymbol{C}t)\boldsymbol{D}\boldsymbol{e}dt \\ &\quad + \int_{t_0}^{\infty} \{P_1 t_0 + P_2(\mathrm{E}[D] + \mathrm{E}[S])\} \exp(\boldsymbol{C}t)\boldsymbol{D}\boldsymbol{e}dt \end{aligned}$$

$$= P_1(-\boldsymbol{C})^{-2}\boldsymbol{De} + P_2\mathrm{E}[S](-\boldsymbol{C})^{-1}\boldsymbol{De}$$

$$+ \{P_2\mathrm{E}[D]\boldsymbol{I} - P_1(-\boldsymbol{C})^{-1}\}\exp(\boldsymbol{C}t_0)(-\boldsymbol{C})^{-1}\boldsymbol{De}. \quad (14)$$

From the renewal reward argument on the Markov renewal process [14], the steady-state power consumption is given by

$$C_{ss}(t_0) = \lim_{t\to\infty} \frac{\mathrm{E}[\text{the amount of consuming electricity during } [0,t)]}{t}$$

$$= \frac{\boldsymbol{\pi}(t_0)\boldsymbol{\xi}(t_0)}{\boldsymbol{\pi}(t_0)\boldsymbol{\tau}(t_0)}. \quad (15)$$

(ii) Throughput

The throughput for the network device is defined as

$$T_{ss}(t_0) = \lim_{t\to\infty} \frac{\mathrm{E}[\text{the number of achieved tasks during } [0,t)]}{t}. \quad (16)$$

Similar to the steady-state power consumption, since the network device always processes only one task requested during one cycle, the throughput can be derived as

$$T_{ss}(t_0) = \frac{1}{\boldsymbol{\pi}(t_0)\boldsymbol{\tau}(t_0)}. \quad (17)$$

4 Performance Evaluation

In this section, we investigate the effectiveness of shutdown policy on the network device under the performance measures. In particular, we consider the following four kinds of arrival streams:

Case 1: Poisson process with the parameters:

$$\boldsymbol{C} = -\lambda, \quad \boldsymbol{D} = \lambda. \quad (18)$$

Case 2: IPP with 2-phases

$$\boldsymbol{C} = \begin{pmatrix} -\mu_1 & \mu_1 \\ \mu_2 & -(\mu_2+\lambda) \end{pmatrix}, \quad \boldsymbol{D} = \begin{pmatrix} 0 & 0 \\ 0 & \lambda \end{pmatrix}. \quad (19)$$

Case 3: IPP with 2-phases and the first arrival causes succeeding requests

$$\boldsymbol{C} = \begin{pmatrix} -\mu_1 & 0 \\ \mu_2 & -(\mu_2+\lambda) \end{pmatrix}, \quad \boldsymbol{D} = \begin{pmatrix} 0 & \mu_1 \\ 0 & \lambda \end{pmatrix}. \quad (20)$$

Case 4: Convolutional of two IPPs in Case 3

$$\boldsymbol{C} = \begin{pmatrix} -2\mu_1 & 0 & 0 & 0 \\ \mu_2 & -(\mu_1+\mu_2+\lambda) & 0 & 0 \\ \mu_2 & 0 & -(\mu_1+\mu_2+\lambda) & 0 \\ 0 & \mu_2 & \mu_2 & -2(\mu_2+\lambda) \end{pmatrix}, \quad (21)$$

$$\boldsymbol{D} = \begin{pmatrix} 0 & \mu_1 & \mu_1 & 0 \\ 0 & \lambda & 0 & \mu_1 \\ 0 & 0 & \lambda & \mu_1 \\ 0 & 0 & 0 & 2\lambda \end{pmatrix}. \tag{22}$$

As shown in Section 3, the arrival stream in Case 3 represents the situation where a communicative session starts and terminates at the rates μ_1 and μ_2, respectively, and where the requests occur at the rate λ while the session is alive. In Case 4, we consider the convolutional of such two arrival streams. Also, it is assumed that both communication time and delay time obey the Gamma distributions with respective shape and scale parameters; (m_s, η_s) and (m_d, η_d), *i.e.*,

$$F(t) = \int_0^t \frac{\eta_s^{m_s} u^{m_s-1} e^{-\eta_s u}}{\Gamma(m_s)} du, \quad H(t) = \int_0^t \frac{\eta_d^{m_d} u^{m_d-1} e^{-\eta_d u}}{\Gamma(m_d)} du. \tag{23}$$

Under these assumptions, the mean times of communication and delay are given by

$$\mathrm{E}[S] = \frac{m_s}{\eta_s}, \quad \mathrm{E}[D] = \frac{m_d}{\eta_d}. \tag{24}$$

If PM does not perform the shutdown, then the traffic intensity is given by

$$\rho = \boldsymbol{\pi}_0 \boldsymbol{D} \boldsymbol{e} \mathrm{E}[S]. \tag{25}$$

Note that this is not an actual traffic intensity, because the additional arrivals are canceled while the network device communicates with one device, but is a appropriate measure to evaluate the busyness of network device. In this paper, we set the above traffic intensity as $\rho = 0.3, 0.5$ by adjusting the parameter λ in each case.

The basic parameters are set as follows.

$$\rho = 0.3, 0.5, \quad \mu_1 = 0.05, \quad \mu_2 = 0.05, \quad m_s = 2.0, \quad \mathrm{E}[S] = 1.0,$$
$$m_d = 2.0, \quad \mathrm{E}[D] = 1.0, \quad P_1 = 1.0, \quad P_2 = 10.0$$

Figure 2 shows the behavior of steady-state power consumption and throughput. Especially, we illustrate the dependence of each measure on varying shutdown timing. From Fig. 2, except for Case 1, *i.e.*, Poisson arrival stream case, there is a unique optimal shutdown timing which minimizes the steady-state power consumption. Also, the minimum expected power consumptions in Case 2 and Case 3 are lower than those in the other cases, so that the effectiveness of power-saving in Case 2 and Case 3 is relatively higher. Next, we focus on the throughput. In this result, the processing performance of network device becomes higher as the shutdown timing is longer. Although the traffic intensities are fixed as 0.3 in all the cases, the throughput in Case 1 is better than the other cases. In other words, the evaluation of performance is considerably optimistic. This is because the Poisson arrival stream has no correlation of arrivals. Figure 3 shows the results in the case of $\rho = 0.5$. For these results, we can make similar observations to the case of $\rho = 0.3$. Thus the optimal shutdown timing strongly

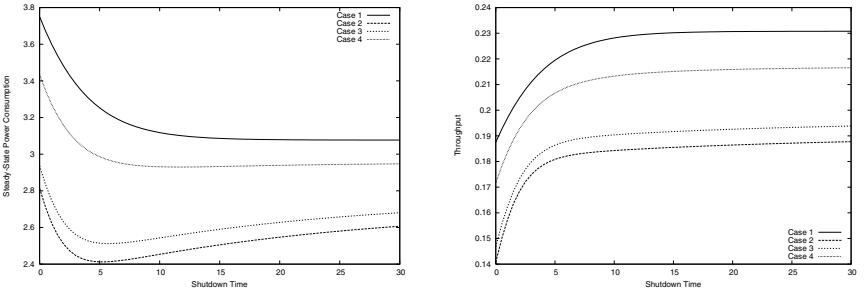


Fig. 2. Performance measures ($\rho = 0.3$)

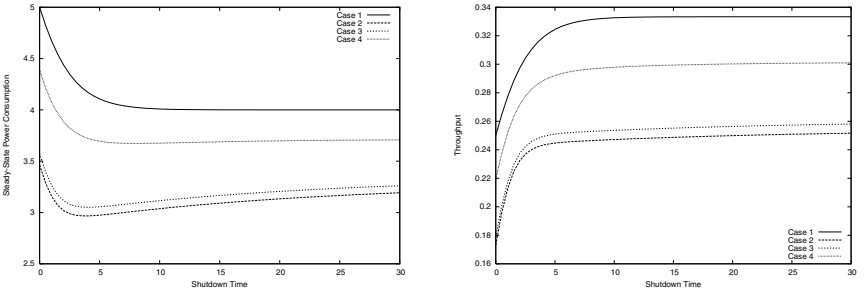


Fig. 3. Performance measures ($\rho = 0.5$)

depends on the correlation of arrivals, rather than the quantities of network traffic. In the situation where the arrival streams are IPPs, the optimal shutdown timing leads to higher power-saving performance in the network device.

5 Conclusion

In this paper, we have considered the power-saving for network devices. Applying the MAP to the arrival stream, we have proposed the DPM model with shutdown policy. Furthermore, based on the DPM model, two performance measures for power-saving and processing have been delivered from the renewal reward theory. In the numerical examples, for four kinds of arrival streams, we have examined the difference on the steady-state power consumption and throughput. As a result, the power-saving performance strongly depends on the correlation of arrival requests. In future, we will develop the DPM model for the network device with communication buffer and the computation algorithm for the optimal shutdown timing.

References

1. L. Benini and G. De Micheli, *Dynamic Power Management: Design Techniques and CAD Tools*, Kluwer Academic Publishers, New York, 1997.
2. L. Benini, A. Bogliolo, G. A. Paloologo, and G. De Micheli, Policy optimization for dynamic power management, *IEEE Trans. on Computer-Aided Design of Circuits Systems*, vol. 18, no. 6, pp. 813–833, 1999.
3. L. Benini and G. De Micheli, System-level power optimization: techniques and tools, *ACM Trans. on Design Automation of Electronic Systems*, vol. 5, no. 2, pp. 115–192, 2000.
4. E. Chung, L. Benini, A. Bogliolo, and G. De Micheli, Dynamic power management for non-stationary service requests, *Proc. 1999 Design Automation and Test in Europe*, pp. 77–81, IEEE Computer Society Press, Los Alamitos, 1999.
5. C. Hwang and A. C. Wu, Predictive system shutdown method for energy saving of event-driven computation, *ACM Trans. on Design Automation of Electronic Systems*, vol. 5, no. 2, pp. 226–241, 2000.
6. Y. Lu and G. De Micheli, Adoptive hard disk power management on personal computers, *Proc. 9th Great Lakes Symp. on VLSI*, pp. 50–53, IEEE Computer Society Press, Los Alamitos, 1999.
7. Y. Lu, T. Šimunić, and G. De Micheli, Software controlled power management, *Proc. 7th Int'l Workshop on Hardware/Software Codesign*, pp. 157–161, ACM Press, New York, 1999.
8. H. Okamura, T. Dohi, and S. Osaki. The phase type approximation for the optimal auto-sleep scheduling. *Mathematical and Computer Modelling*, vol. 38, nos. 11/13, pp. 1391–1398, 2003.
9. H. Okamura, T. Dohi, and S. Osaki. A structural approximation method to generate the optimal auto-sleep schedule for a computer system. *Computers and Mathematics with Applications*, vol. 46, no. 7, pp. 1103–1110, 2003.
10. Q. Qiu and M. Pedram, Dynamic power management based on continuous-time Markov decision processes, *Proc. 36th Design Automation Conference*, pp. 555–561, IEEE Computer Society Press, Los Alamitos, 1999.
11. H. Sandoh, H. Hirakoshi, and H. Kawai, An optimal time to sleep for an auto-sleep system, *Computers & Operations Research*, vol. 23, pp. 221–227, 1996.
12. M. Srivastava, A. Chandrakasan, and B. Brodersen, Predictive system shutdown and other architectural techniques for energy efficient programmable computation, *IEEE Trans. on Very Large Scale Integrated Systems*, vol. 4, no. 1, pp. 42–55, 1996.
13. M. F. Neuts, *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach*, Johns Hopkins University Press, Baltimore, 1981.
14. D. R. Cox, *Renewal theory*, John Wiley & Sons Inc., London, 1962.

An Energy Aware, Cluster-Based Routing Algorithm for Wireless Sensor Networks*

Jyh-Huei Chang** and Rong-Hong Jan

Department of Computer and Information Science,
National Chiao Tung University, Hsinchu, 30050, Taiwan
jhchang@cis.nctu.edu.tw

Abstract. Cluster-based routing protocol has special advantages to enhance scalability and efficiency of the routing protocol. The technique to network clustering that maximizes the network lifetime is an important topic of research in wireless sensor networks. In this paper, we present an energy aware cluster-based routing algorithm (ECRA) for the wireless sensor networks such that the network lifetime can be maximized. ECRA randomly selects some nodes as cluster-heads to construct Voronoi diagram and rotates the intra-cluster nodes to distribute the energy load to all sensors in the network. The simulations show that ECRA algorithm outperforms than direct communication, static clustering and LEACH. The system lifetime of ECRA is approximately 2 times than LEACH. This is because the ECRA rotates intra-cluster-heads to balance the load to all nodes in the sensor networks.

1 Introduction

The technologies of micro-electro-mechanical systems (MEMS) have quickly progressed. These MEMS technologies combine with the wireless communication make it possible to deploy low-cost and low-power sensor network. Many civil and military applications of wireless sensor network such as environmental monitors, informational gathers, battlefields monitors and detecting ambient conditions such as sound, temperature or light. The sensor device has limited energy and transmitting range. Thus, finding the best use for this limited resources is an important and key research topic in wireless sensor network.

Many studies have focused on saving energy in different ways such as reducing the power spent on the modulation circuits[1], or managing the power usage on the MAC layer of sensor nodes [2, 3]. However, these schemes cared of the individual device that is not enough in wireless sensor networks. Since sensor nodes have limited transmitting ranges so that only a few nodes can communicate directly with the sink node. In most cases, the sensor nodes gather sensing

* This research was supported in part by the National Science Council, Taiwan, ROC, under grant NSC 94-2219-E-009-005 and NSC94-2752-E-009-005-PAE and in part by the Intel.

** Corresponding Author. Fax: 886-3-5721490

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 255–266, 2005.
© IFIP International Federation for Information Processing 2005

data which must be forwarded by other node to the sink node. These relaying operations consume too much energy such that the relay nodes will rapidly expend much of their energy. Therefore, developing an energy-aware and load-balanced routing algorithms to maximize the network lifetime has become an important research issue in wireless sensor network.

A large number of routing protocols for wireless sensor networks have been proposed. In the fixed path schemes [4, 5], sensor nodes which are in a fixed path will consume much energy and get exhausted rapidly. This is because they provide relaying service. Flooding scheme consumes too much energy for relaying duplicate packets. Source routing schemes [6, 7] improved the drawback of flooding scheme. However, they can not operate well when the number of hops between source and sink is large. Regarding energy-aware, multi-path routing schemes [8, 9, 10] have the advantage sharing the energy between all sensor in the wireless networks. Nevertheless, the disadvantage of multi-path routing schemes [8, 9] is that the sensor nodes only keep a local view of energy usage and the nodes in the network can not have even traffic dispatch.

There are many studies focus on cluster-based energy-efficient routing protocol for wireless sensor networks [11-19]. The Low Energy Adaptive Clustering Hierarchy (LEACH) [11, 12] randomly selects some nodes as cluster-heads and rotates the cluster-head to distribute the load to all sensors in the wireless sensor networks. It's performance is better than direct communication and static clustering routing protocols. However, the issues of LEACH are as follows: First, if the coverage of cluster-head is too small area, then some cluster-heads will not have any member in its cluster. Second, LEACH has a long latency for the base station to receive the data from the cluster-heads. A chain-based enhanced protocol over LEACH was proposed [13]. It allows nodes need only communicate with their closest neighbors and take turns in communicating with the sink. This routing scheme can increases the operating lifetime of each node and reduces bandwidth consumed. However, this scheme assumes that each node maintains a full database about the position of all other nodes, but the method is not outlined. In [14, 15, 16], a family of adaptive protocols that address the classic flooding using negotiation and resource adaptation was presented. Nevertheless, the high degree nodes of this scheme consume more energy and reduce network lifetime. The above cluster-based schemes require global cluster-heads rotation. This cluster-heads re-election increases many processing and communications overhead. Thus, this paper presents a energy-aware routing scheme to reduce the overhead of cluster-heads rotation for cluster-based wireless sensor networks.

Figure 1 shows an example of wireless sensor network with cluster-based routing scheme. In Figure 1, each cluster has one cluster-head. The non-cluster-head nodes transmit its sensing data to cluster-head which forwards the aggregated data to the sink node. The use of cluster leverages the benefits of small transmit distance for most nodes. The cluster-head acts as a fusion point to aggregate the sensing data such that the amount of data that is actually transmitted to the sink node is reduced [11, 12]. Thus, network clustering can contribute to system

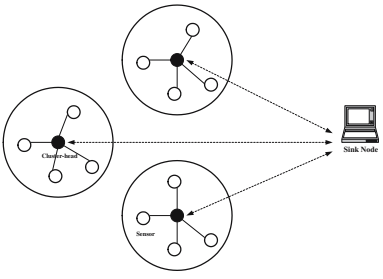


Fig. 1. Wireless sensor network organized in clusters

lifetime and energy efficiency. Cluster-based routing protocol has special advantages to enhance scalability and efficiency of the routing protocol such as reduces the routing complexity [20], reduces the complexity of location management [21], and improves the power control procedure [22]. This technique to network clustering that maximize the network lifetime is an important topic of research in wireless sensor networks. In this paper, we present an energy aware cluster-based routing algorithm (ECRA) for the wireless sensor network such that the network lifetime can be maximized. The ECRA algorithm includes three phases: clustering, data transmission and intra-cluster-head rotation. In our work, we assume that the sensors are location aware. The ECRA algorithm randomly selects some nodes as cluster-heads to construct Voronoi diagram and rotates the intra-cluster nodes to distribute the energy load to all sensors in the network. These are the differences from LEACH. Our motivation is to balance the load to all sensors and avoid too many cluster-heads focusing on the small area. Simulation results show that the system lifetime of ECRA is approximately 2 times than LEACH. ECRA achieves as much as a factor of 9 reduction in the energy dissipation compared with direct communication. This is, ECRA gives better performance than other schemes in system lifetime and energy dissipation. This means that ECRA scheme shares the load evenly to all sensor nodes in the wireless sensor network and it can achieve a better lifetime.

The remainder of this paper is organized as follows. In section 2, we describe energy-based localization. Section 3 illustrates the details of ECRA algorithm. The simulation results and performance analysis are shown in section 4. Finally, the conclusions are given in section 5.

2 The Energy Model of ECRA

In our work, the energy model is the same as in [11]. In this energy model, the electronics energy E_{elec} =50 $nJ/$bit to operate the transmitter or receiver circuit. The transmitter amplifier $\epsilon_{amp} = 100\ pJ/bit/m^2$. Equations (1) and (2) are used to calculate the transmission energy, denoted as $E_{Tx}(k, d)$, spend for a k bits message over a distance d.

$$E_{Tx}(k, d) = E_{Tx_elec}(k) + E_{Tx_amp}(k, d) \tag{1}$$

$$= E_{elec} * k + \epsilon_{amp} * k * d^2 \tag{2}$$

and to receive this message, the energy expends:

$$E_{Rx}(k) = E_{Rx_elec}(k) = k * E_{elec} \tag{3}$$

Where E_{Tx_elec} is the energy dissipation of transmitter electronics and E_{Rx_elec} is energy dissipation of receiver electronics. E_{Tx_amp} is the energy of transmitter amplifier. Assume that $E_{Tx_elec} = E_{Rx_elec} = E_{elec}$. From equation (3), receiving data is also a high overhead procedure. Thus, the number of transmission and receiving operations must be minimized to reduce the energy dissipation. We also assume that the radio channel is symmetric such that the energy required to transmit a message from node i to node j is the same as the energy required to transmit a message from node j to node i for a given signal to noise ratio.

3 The Details of ECRA

The ECRA algorithm includes the following three phases: clustering, data transmission and intra-cluster-head rotation. The details of the algorithm are given as follows.

Phase 1: Clustering

Initially, the base station randomly selects 5 percentage[1] of total sensor nodes as the cluster heads and informs these nodes that they will become cluster-heads. The set of cluster-heads is denoted as S_c. CH_i represents the cluster-head of cluster i. The set of nodes is denoted as S_n. N_{ij} represents node j of cluster i (It is shown in Figure 2). In this wireless sensor networks, the locations of the sensors are static such that the intersection area can be determined by Voronoi diagram.

We define the Voronoi diagram as follows [23, 24]. Assume that $P = \{p_1, p_2, ..., p_n\}$ stands for the set of points in the n-dimensional space. Their location vectors of p_i and p_j is denoted as l_i and l_j. $l_i \neq l_j$, $\forall i \neq j$. The region that is given by $S(p_i) = \{l \mid \| l - l_i \| \leq \| l - l_j \|, \forall i \neq j\}$ that is called the Voronoi region. $V(P) = \bigcup_{1 \leq i \leq N} S(p_i)$ is called the Voronoi diagram. The half-space is bounded by the hyperplane that is perpendicular bisector of a segment. $S(p_i)$ is the intersection area of these half-space. Therefore, the Voronoi region can defined as follow. $S(p_i) = \bigcap_{1 \leq j \leq N, j \neq i} h(p_i, p_j)$. Where, $h(p_i, p_j)$ stands for the half-space containing p_i. In our ECRA scheme, the Voronoi diagram is applied for network clustering. The base station constructs Voronoi diagram to cover the sensing field and selects a sensor node as cluster-head for each Voronoi cell. The base station also informs each cluster-head which nodes are its members. The cluster-head broadcasts an advertisement message to their members. After each node known which cluster it belongs, then it informs the cluster-head that it will be a member of the cluster using a CSMA MAC protocol. During this time, all cluster-heads must keep in active.

[1] This parameter has minimal normalized total energy dissipation [11].

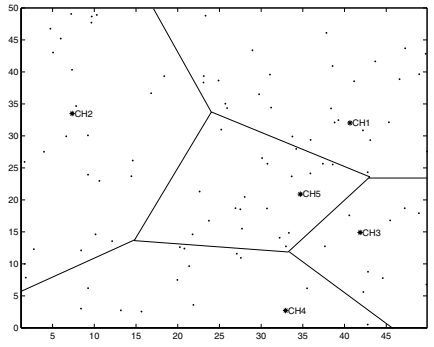


Fig. 2. Voronoi diagram of the 100-node random sensor network

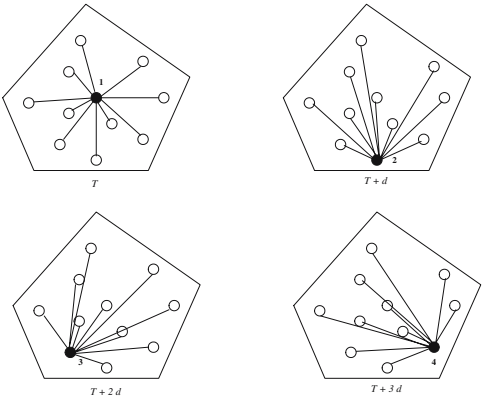


Fig. 3. An example of intra-cluster-head rotation. Where T is the current round and $T + d$ is the next round, and so on.

Phase 2: Data transmission

When the clusters are created such that the data transmission can begin. The nodes use single hop to communicate with their cluster-heads, and the cluster-heads also use single hop to communicate with the base station. Each node has M bits messages to transmit. The non-cluster-head node can turned-off until its allocated transmission time to minimize the energy dissipation. When all data from the nodes has been received, the cluster-head aggregates the total data into a single message to reduce the amount of information that transmitted to the base station.

Phase 3: Intra-cluster-head rotation

When a round is end, then rotate the cluster-head within the same cluster based on a parameter called O_{ij} which considers the distance from node j in cluster i to base station and the residual energy of each node. The distance from node j in cluster i to base station is denoted as d_{ij}.

Table 1. The pseudo code of ECRA algorithm

```
begin Clustering
  Randomly select cluster-heads
   for (every S_c)
     Construct Voronio diagram
   end for
end Clustering
begin Data transmission
while ( data size < M )
      for ( every S_c )
         for ( every N_ij )
           Transmit data
         end for
      end for
end while
end Data transmission
begin Intra-cluster-head rotation
while ( data size = M )
      for ( every S_c )
         for ( every N_ij )
            Calculate O_ij
            CH_i,new = N_ij
           j = j + 1
         end for
      end for
end while
end Intra-cluster-head rotation
```

$$d_{ij} = \sqrt{(X_{ij} - X_{bs})^2 + (Y_{ij} - Y_{bs})^2} \tag{4}$$

$$E_{new} = E_{old} - E_{expend} \tag{5}$$

$$O_{ij} = \frac{E_{new}}{d_{ij}} \tag{6}$$

Where the position of node j in cluster i is at (X_{ij}, Y_{ij}) and the position of base station is at (X_{bs}, Y_{bs}). In equation (5), E_{old} is the residual energy of each node at current round and E_{new} is the residual energy of each node at next round. E_{expend} is the energy expend of each node in the current round. From equation (6), O_{ij} is a function of d_{ij} and E_{new}. The node with the maximal value of O_{ij} will become a cluster-head at the next round (It is shown in Figure 3). When all data are received in current round. The base station informs the cluster-head of current round that the other node will become the new cluster-head $CH_{i,new}$ at next round. The base station has the location of each node and it can calculate

the value of O_{ij}. When the current round is end, the cluster-head will rotate to the other node that informed by the base station. Then, the new cluster-head begins to advertise using the same method as phase 1. The pseudo code of ECRA algorithm is shown in Table 1.

4 Simulation Results

4.1 Performance Metrics and Environment Setup

This section presents the performance analysis of the ECRA algorithm. The metrics for performance are given as follows.

1) The lifetime for first node to die (FND): FND is defined as the time that the first node run out of energy. The non-cluster-head nodes transmit their sensing data to cluster-head. The cluster-heads returned their aggregated data to the sink periodically. We use the number of *rounds* to represent the network lifetime of FND. A round is defined as all nodes in the wireless network that finish returning their gathered data to the sink. The time interval between two rounds is assumed to be large enough for last node to return its sensing data.
2) The lifetime for last node to die (LND): LND is defined as the time that last node run out of energy and the network crashed. We also use the number of *rounds* to represent the network lifetime of LND.
3) The total energy dissipation: This value is defined as the energy dissipation for all nodes that finish returning their gathered data.

Four different sizes of deploying regions were simulated. There are $50\times50\ m^2$, $100 \times 100\ m^2$, $150 \times 150\ m^2$ and $200 \times 200\ m^2$. In each region, 100 nodes was deployed by uniform distribution. Assume that the energy model is the same as in [11]. The electronics energy E_{elec} =50 nJ/bit. $\epsilon_{amp} = 100\ pJ/bit/m^2$. The energy of data aggregation is 5 nJ/bit/message. There is a 100 - node random network with 5 percentage cluster heads. The sink node was located at the position (x = 25, y = -100). Each sensor has 2000 bits data packet to the base station during each round. We evaluate three performance metrics of ECRA and compare with other schemes.

4.2 Numerical Results

Comparisons of the three performance metrics were made for four schemes: the direct communication (DC), static clustering (SC), LEACH and ECRA. These performance metrics includes the FND, LND and total energy dissipation in the system that will illustrate as follows.

1) The lifetime of FND under different initial energy

Figure 4 shows that the lifetime of ECRA in FND is approximately 1.8 times than LEACH, 13 times than direct communication and 29 times than static

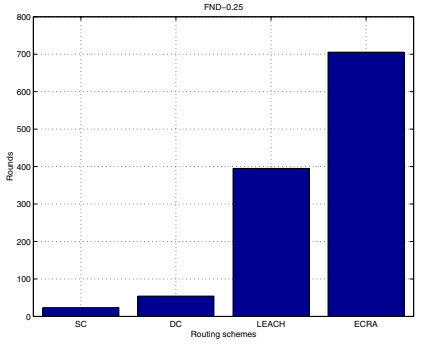


Fig. 4. The lifetime of first node died (FND) under different methods. The initial energy of each sensor is 0.25.

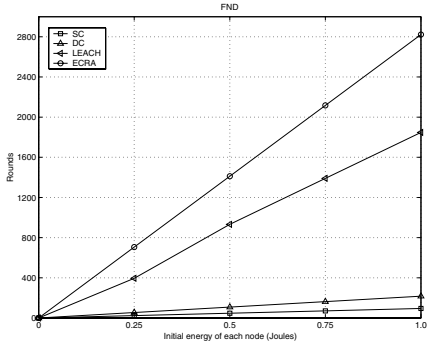


Fig. 5. The lifetime of first node died (FND) using different amounts of initial energy for the sensors

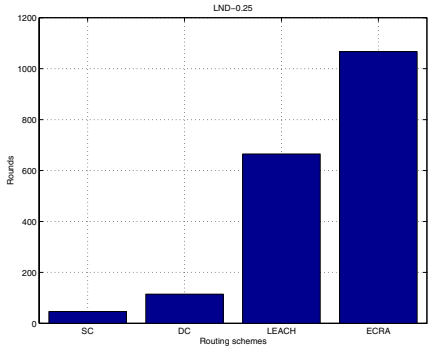


Fig. 6. The lifetime of last node died (LND) under different methods. The initial energy of each sensor is 0.25.

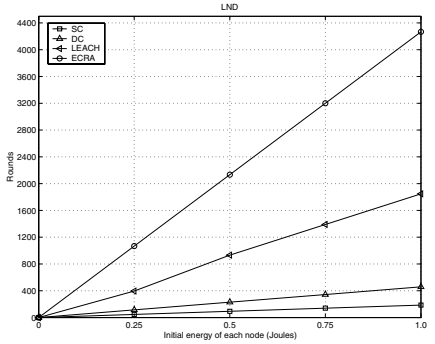


Fig. 7. The lifetime of last node died (LND) using different amounts of initial energy for the sensors

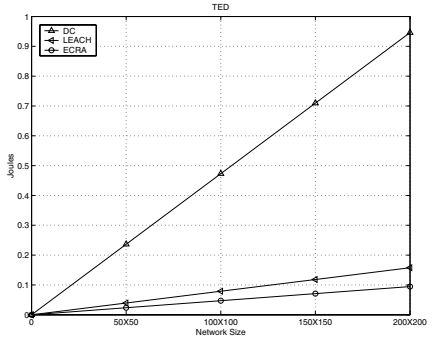


Fig. 8. Total energy dissipation (TED) using direct communication, LEACH and ECRA. The messages are 2000 bits.

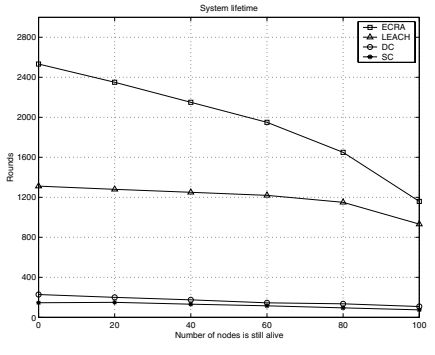


Fig. 9. System lifetime using direct communication, static clustering, LEACH and ECRA with 0.5 J / node

clustering if the initial energy of each sensor is 0.25 J. Figure 5 shows that the lifetime of FND under different methods with different initial energy of each node. Overall, the lifetime of FND is increasing when the initial energy of each sensor is greater. This figure shows that the lifetime of FND in ECRA is approximately up to 2 times than LEACH, over 13 times than direct communication and static clustering. That is, ECRA gives better performance than DC, SC, and LEACH in the lifetime of FND.

2) The lifetime of LND under different initial energy

Figure 6 shows that the lifetime of ECRA in LND is approximately 1.7 times than LEACH, 10 times than direct communication and 23 times than static clustering if the initial energy of each sensor is 0.25 J. Figure 7 shows that the lifetime of LND under different methods with different initial energy of each node. Overall, the lifetime of LND is increasing when the initial energy of each sensor becomes large. This figure shows that the lifetime of LND in ECRA is approximately up to 2 times than LEACH, over 18 times than direct communication and static clustering. The results show, ECRA gives better performance than DC, SC, and LEACH in the lifetime of LND. From Figure 5 and Figure 7, note that a scheme shares the load evenly to all sensor nodes in the network, it can achieve a better lifetime.

3) Total energy dissipation under different network diameter

Figure 8 shows that ECRA achieves as much as a factor of 9 reduction in the energy dissipation compared with direct communication. This means that the use of cluster leverages the benefits of small transmit distance for most nodes and distributes the energy among the sensor nodes in the network can reduce total energy dissipation.

Figure 9 shows that the system lifetime is decreasing when the the number of rounds is greater. From this figure shows that the system lifetime of ECRA is approximately 2 times than LEACH. According to above analysis, our ECRA algorithm has better performance than other schemes in system lifetime and energy dissipation. These simulation results show that ECRA has the advantages of load-balancing and energy-saving.

5 Conclusions

Cluster-based routing protocol has special advantages to enhance scalability and efficiency of the routing protocol. This paper presents an energy aware cluster-based routing algorithm for wireless sensor networks. Compared with direct communication, static clustering and LEACH, our ECRA scheme can achieve longer lifetime. This is because the ECRA rotates intra-cluster-heads to balance the load to all nodes in the sensor networks. The numerical results show that ECRA gives better performance in load-balancing and energy-saving than other schemes. We are confident that ECRA is an efficient and useful algorithm for further wireless ad hoc sensor networks.

References

[1] C. Chien, I. Elgorriaga, and C. McConaghy: Low-Power Direct-Sequence Spread-Spectrum Modem Architecture for Distributed Wireless Sensor Networks. Proceedings of the 6th ACM International Symposium on Low Power Electronics and Design (ISLPED'03). (2003) 251-254

[2] A. Sinha and A. Chandrakasan: Dynamic Power Management in Wireless Sensor Networks. IEEE Design and Test of Computers. (2001) 62-74

[3] Y. Wei, J. Heideman, and D. Estrin: An Energy-Efficient MAC Protocol for Wireless Sensor Networks. Proceedings of 21th Annual Joint Conference of the IEEE Computer and Communications Societies. (2002) 1567-1576

[4] J. Moy: OSPF Version 2. RFC 2178. Internet Engineering Task Fource (1997)

[5] J. Moy: OSPF : Anatomy of An Internet Routing Protocol, Addison-Wesley (1998)

[6] A. Boukerche, X. Cheng, and J. Linus: Energy-Aware Data-Centric Routing in Microsensor Networks. Proceedings of the 6th ACM International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems. (2003) 459-468

[7] R. C. Shah and J. M. Rabaey: Energy Aware Routing for Low Energy Ad Hoc Sensor Networks. Proceedings of Wireless Communications and Networking Conference. (2001)

[8] X. Hong, M. Gerla, R. Bagrodia, J. K. Taek, P. Estabrook and P. Guangy: The Mars Sensor Network: Efficient, Power Aware Communications. Proceedings of Military Communications Conferences (MILCOM'01). (2001) 418-422

[9] X. Hong, M. Gerla, W. Hanbiao, and L. Clare: Load Balanced, Energy-Aware Communications for Mars Semsor Networks. Proceedings of IEEE Aerospae Conference. (2002) 1109-1115

[10] K. Sohrabi, J. Pottie: Protocols for Self-Organization of a wireless Sensor Network. IEEE Personal Communications. (2000) 16-27

[11] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan: Energy-Efficient Communication Protocol for Wireless Microsensor Networks. Proceedings of HICSS. (2000) 1-10

[12] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan: An Application-Specific Protocol Architecture for Wireless Microsensor Networks. IEEE Transactions on Wireless Communications. (2002)

[13] S. Lindsey and C. Raghavendra: PEGASIS: Power-Efficient Gathering in Sensor Information Systems. IEEE Aerospace Conference Procedings. (2002) 1125-1130

[14] W. R. Heinzelman, J. Kulik, and H. Balakrishnan: Adaptive Protocols for Information Dissemination in Wireless Sensor Networks. Proceedings of the ACM MobiCom'99. (1999) 174-185

[15] J. Kulik, W. R. Heinzelman, and H. Balakrishnan: Negotiation-based Protocols for Disseminating Information in Wireless Sensor Networks. Wirelwss Networks. (2002) 521-534

[16] C. Chiasserini, I. Chlamtac, P. Monti, and A. Nucci: An Energy-Efficient Method for Nodes Assignment in Cluster-Based Ad Hoc Networks. ACM Wireless Networks. (2004) 223-231

[17] S. Bandyopadhyay and E. J. Coyle: An Energy Efficient Hierarchical Clustering Algorithm for Wireless Sensor Networks. IEEE INFOCOM. (2003) 1713-1723

[18] M. Younis, M. Youssef, and K. Arisha: Energy-Aware Routing in Cluster-based Sensor Networks. Proceedings of the 10th IEEE International Symposium. (2002) 129 -136

[19] V. Mhatre and C. Rosenberg: Homogeneous vs Homogeneous Clustered Sensor Networks: A Comparative Study. IEEE International Conference on Communications (ICC'04). (2004)
[20] A. Iwata, C.-C. Chiang, G. Pei, M. Gerla and T.-W. Chen: Scalable Routing Strategies for Ad Hoc Wireless Networks. IEEE JSAC. (1999) 1369-1379
[21] W. Chen, N. Jain, and S. Singh: ANMP: Ad Hoc Network Management Protocol. IEEE JSAC. (1999) 1506-1531
[22] T. Kwon and M. Gerla: Clustering with Power Control. Proceedings of Military Communications Conferences (MILCOM'99). (1999) 1424-1428
[23] K. Mulmuley: Computational Geometry, Prentice Hall (1994)
[24] F. Aurenhammer: Voronoi Diagrams - A Survey of A Fundamental Geometric Data Structure. ACM Computing Surveys 23. (1991) 345-405

Energy-Constrained Prefetching Optimization in Embedded Applications*

Juan Chen, Yong Dong, Hui-zhan Yi, and Xue-jun Yang

School of Computer, National University of Defense Technology,
Changsha 410073, China
juanchen@nudt.edu.cn, luckpeople@163.com
huizhanyi@nudt.edu.cn

Abstract. In energy-constrained settings, most low-power compiler optimization techniques take the approach of minimizing the energy consumption while meeting no performance loss. However, it is possible that the available *energy budget* is not sufficient to meet the optimal performance objective. To limit energy consumption within a given energy budget, energy-constrained optimization approach is more significant. In this paper, we present an energy-constrained prefetching optimization approach through which memory or CPU stalls (caused by too early or too late prefetching) can be reduced so that energy budget is met. Optimal performance objective is achieved under a given energy budget. We evaluate the effectiveness of our energy-constrained prefetching optimization approach through simulations.

Keywords: software prefetching, energy-constrained, DVS, embedded applications.

1 Introduction

With the development of wireless, portable and embedded devices, power-aware systems have moved to the forefront of computer research in recent years, among which **Dynamic Voltage Scaling** (**DVS**) is major low-power techniques [1][2][3][4][5].

Many low-power techniques focus on minimizing the energy consumption with performance constraints. In this paper, we look at the problem from the other perspective: we consider optimized application with energy constraint (limitation). We assume that a limited energy budget (E_{budget}) is provided and it is usually less than the energy bound (E_{bound}) for the optimal performance objective. Such a situation is significant when the computing device depends on the battery power supply, such as military, space, and disaster recovery missions, etc. In this paper, we are concerned

* This work was supported by the National High Technology Development 863 Program of China under Grant No. 2002AA1Z2101 and No. 2004AA1Z2210.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 267 – 280, 2005.
© IFIP International Federation for Information Processing 2005

with energy-constrained prefetching optimization. With our approach, applications optimized with prefetching reach the optimal performance within the available energy budget. Clearly, the compiler can make use of well-known power-aware scheduling techniques (such as DVS) for energy savings. In scarce-energy settings where $E_{budget} < E_{bound}$, we have a serious problem: we have to sacrifice some prefetching optimization performance.

Software prefetching improves performance by effectively hiding memory access latency through overlapping memory access with computation [6]. However, software prefetching does not always implement perfect data prefetching because it requires prefetch instructions are inserted at the right places. Prefetching too early or too late may yield memory or CPU stalls. By reducing memory frequency or CPU frequency or both, we not only can reduce these stalls but also can limit energy consumption within a given energy budget.

There usually exist two imperfect prefetch optimization cases: **CPU-bound case** and **memory-bound case**. In CPU-bound case, prefetching operation is too early or memory access latency is too short so that the prefetched data is provided before actually being used. Although memory access latency is completely hidden, prefetching too early makes memory access completes ahead of the actual data access. We refer to this time interval as *memory stalls*. The second case is memory-bound case, where memory access dominates the whole time and CPU stalls cannot be avoided due to prefetching too late or too long memory access latency. Figure 1 illustrates the behavior of prefetching.

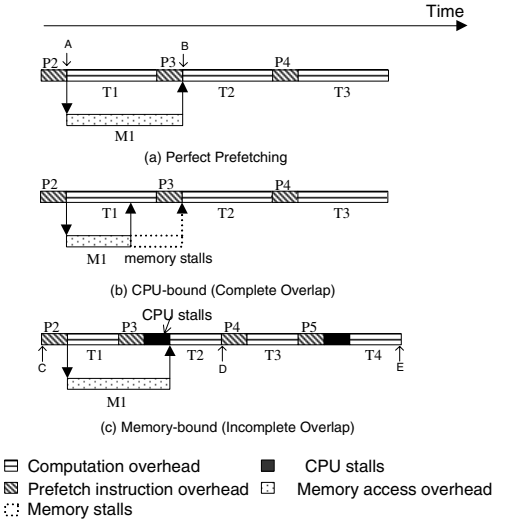


Fig. 1. Prefetching Optimization. P2, P3, P4, P5 represent prefetch instrution overhead for the second, third, fourth and fifth iteration, respectively. M1 represents memory access overhead caused by prefetching. T1, T2, T3 and T4 represent computation overhead in the first, second, third and fourth iteration. Note: here iteration denotes the outmost iteration.

With DVS, we utilize CPU stalls and memory stalls to reduce memory frequency or CPU frequency or both so as to meet scarce energy budget. In CPU-bound case, the major method is to adjust memory frequency to eliminate memory stalls combined with small CPU frequency scaling. In memory-bound case, the major approach is to reduce CPU frequency to eliminate CPU stalls combined with small memory frequency scaling. Furthermore, it is necessary to adjust both of their frequencies instead of one of them for the optimal performance under energy constraint.

In this research effort, we introduce the notion *time* to characterize the performance benefits—*time* denotes the program execution time with software prefetching optimization. Optimal performance objective is minimizing *time*.

When energy is scarce, DVS capability (DVS for memory and CPU) guarantees that maximum performance objective will be met while staying within a given energy budget.

We simulate the impact of different parameters on our optimization. Experimental results show our approach is effective in implementing energy-constrained performance optimization.

The remainder of this paper is organized as follows. Section 2 reviews the related works. In section 3, we formulate an energy-constrained software prefetching problem to an optimization problem. Section 4 gives compilation strategy. Section 5 provides experimental results and analysis. Section 6 gives the conclusions.

2 Related Works

There are a lot of literatures on software prefetching techniques [6][7][8]. Todd [6] provided a comprehensive analysis on software prefetching. Ricardo Bianchini et al. [9] presented analytical models of the performance benefits of multithreading and prefetching. Our work built the energy model of prefetching.

DVS for CPU has been widely researched [1][2][3][4][10][11]. Furthermore, currently power-aware memory has been concerned by some researches besides DVS of CPU [5][12]. Pouwelse [12] alluded to the problem that the high cost of memory may dominate the overall power consumption of a system so that effective DVS of the CPU delivers marginal benefit. Xiaobo Fan et al. [5] exploited a positive synergistic effect between DVS and power-aware memories that can be transferred into lower power states. They believe that the best frequency/voltage for minimizing power consumption should be obtained by including memory power in the decision (The energy savings from DVS alone with standard memory are only 39% compared with energy savings of 89% by consistent method).

We exploited memory stalls and CPU stalls existing in imperfect prefetching and utilized them to reduce CPU or memory frequency/voltage for energy savings.

3 Energy-Constrained Prefetching Optimization

The prototypical loop that we optimize with prefetching looks like:

```
for ( i = 0 ; i < N ; i ++)  Compute ( i ) ;
```

After software prefetching optimization, the above loop is changed into:

```
// iteration 0, prefetch b blocks
prefetch ( 0, b );
for ( i = 0 ; i < N - step ; i += step ) {
    // prefetch b blocks for iterations i+step to i+2*step-1
    prefetch  ( i + step , b);
    // compute iterations i to i + step - 1
    for ( j = 0 ; j < step ; j ++ )  Compute ( i * step + j );
}
for ( j = 0 ; j < step ; j++ )  Compute ( i * step + j ) ;
```

In the above prefetching loop, several prefetch instructions are inserted. The major parameters involved in modeling the energy behavior of the above prefetching loop are the number of the cache blocks prefetched for each prefetching instruction, b; the number of prefetching instructions per iteration, N_b; the energy overhead of each prefetched cache block, E_b; the time overhead of each prefetched cache block, C_p; the total latency of cache misses per iteration, C_m; and the computation between two consecutive prefetch instructions, C_c. Table 1 summarizes these parameters.

Prefetching allows greater flexibility when trying to overlap memory access and computation, since the compiler or user can schedule the prefetches according to the memory access latency and the amount of work per iteration of each loop in the program. This intelligent scheduling of prefetches adjusts b and the iteration *step* size appropriately as the above code segment shows.

The optimal number of blocks to prefetch at a time depends on the available cache space; blocks prefetched into the cache may be replaced before being used. Another limitation is the number of prefetched blocks that can fit in a prefetch/transaction buffer. With these constraints, the amount of useful work that can be overlapped with memory access may be insufficient to hide memory latency completely. To analyze this problem, we simplify it as Figure 1 shows.

Figure 1 illustrates the behavior of the prefetching loop. The execution alternates between prefetch instruction and computation intervals. Each prefetch accessing cache blocks are to be used one computation block ahead. While it is possible to have loops that prefetch more than one computation block ahead, we can always transform such loops into an equivalent loop that prefetches a single computation block ahead.

In Figure 1(a), **P2** denotes prefetch instruction operation, where calculates the data address to be prefetched (b, N_b, E_b are involved). **M1** executes memory access operations caused by prefetch instruction (C_m, f_m are involved). **T2** accesses the prefetched data at point **B** (C_c, f_c are involved). Figure 1(a) shows perfect prefetching behavior. Memory accesses caused by prefetch are fully overlapped with computation, where no CPU stall or memory stall exists. If prefetech instructions cannot be inserted at the right places, two kinds of imperfect prefetchings occur as shown in Figure 1(b) and Figure 1(c). In terms of CPU-bound or memory-bound case, we adjust memory frequency f_m and CPU frequency f_c to meet energy budget.

Our energy-constrained prefetching optimization problem is stated as follows: given a loop L optimized with software prefetching and a DVS-enabled CPU and memory system, find the optimal CPU frequency f_c and memory frequency f_m, then the performance objective (time) is minimized while meeting a given energy budget.

3.1 Energy Constraint

The total energy consumption E_{total} consists of three parts: E_p, E_c, and E_m. That is,

$$E_{total} = E_p + E_c + E_m$$

where E_p represents the energy consumption of prefetching instructions. We assume the number of prefetched cache block per iteration is B, which is the product of b and N_b. Thus the energy consumed per loop iteration is $E_b \cdot b \cdot N_b$ and the energy consumption of the prefetching instructions E_p for N loop iterations is:

$$E_p = E_b \cdot b \cdot N_b \cdot N$$

E_c represents the energy consumption of CPU computation. Due to continuously variable CPU frequency, $P_c(f_c)$ is denoted the CPU power dissipation for CPU frequency of f_c. $\frac{C_c}{f_c}$ represents CPU computation time. Then CPU energy consumption during N loop iterations is:

$$E_c = P_c(f_c) \cdot \frac{C_c}{f_c} \cdot N$$

E_m represents memory energy consumption. Due to continuously variable memory frequency, $P_m(f_m)$ represents the memory power dissipation for memory frequency of f_m. $\frac{C_m}{f_m}$ represents memory access time. Then the memory access energy consumption for N iterations is:

$$E_m = P_m(f_m) \cdot \frac{C_m}{f_m} \cdot N$$

Thus E_{total} can be represented by:

$$E_{total} = E_b \cdot b \cdot N_b \cdot N + P_c(f_c) \cdot \frac{C_c}{f_c} \cdot N + P_m(f_m) \cdot \frac{C_m}{f_m} \cdot N$$

Given an energy budget E_{budget}, the following energy constraint must be satisfied:

$$E_b \cdot b \cdot N_b \cdot N + P_c(f_c) \cdot \frac{C_c}{f_c} \cdot N + P_m(f_m) \cdot \frac{C_m}{f_m} \cdot N \le E_{budget} \quad (1)$$

Power characteristics of CMOS is $P = \alpha CV^2 f$ and the relationship between supply voltage V and frequency f is $f \propto \frac{(V - V_{th})^{\beta}}{V}$, where V denotes supply voltage, f denotes clock frequency, P denotes power dissipation, α represents the activity factor, C represents the capacitance, V_{th} represents threshold voltage, and β is a proportional factor between 1 and 2. It is reasonable to assume frequency f is linearly proportion with voltage V so that we can get formula (2).

$$P = \alpha \cdot C \cdot f^3 \tag{2}$$

According to formula (2), we can achieve $P_c(f_c) = \alpha_1 \cdot C_1 \cdot f_c^3$ and $P_m(f_m) = \alpha_2 \cdot C_2 \cdot f_m^3$. So formula (1) is transformed to formula (3).

$$E_b \cdot b \cdot N_b \cdot N + k_1 \cdot f_c^2 \cdot C_c \cdot N + k_2 \cdot f_m^2 \cdot C_m \cdot N \le E_{budget} \tag{3}$$

where k_1 represents $\alpha_1 \cdot C_1$ and k_2 represents $\alpha_2 \cdot C_2$.

3.2 Optimal Performance Objective

In the previous section, we have defined performance objective—*time*. To calculate this value, two cases need to be considered. In CPU-bound case (Figure 1(b)), the total execution time of prefetching loop is calculated by prefetching instruction and CPU computation as follows.

$$T_{total} = N \cdot \left(\frac{b \cdot N_b \cdot C_p}{f_c} + \frac{C_c}{f_c}\right) \quad cond1$$

In memory-bound case, CPU stall occurs every other prefetching operation as Figure 1(c) shows. The time interval from **C** to **D** and the time interval from **D** to **E** are the same. We can conclude that the total execution cycles are *N/2* times as many as the execution cycles during **C** to **D**. Since the execution cycles during **C** to **D** are $\left(\frac{b \cdot N_b \cdot C_p}{f_c} + \frac{C_m}{f_m} + \frac{C_c}{f_c}\right)$, the total execution time is[1]

[1] Here we assume $N \bmod 2 = 0$. If $N \bmod 2 = 1$, $T_{total} = \frac{N+1}{2} \cdot \left(\frac{b \cdot N_b \cdot C_p}{f_c} + \frac{C_m}{f_m} + \frac{C_c}{f_c}\right) - \frac{C_m}{f_m}$. When N is large enough, this little difference caused by N can be ignored.

$$T_{total} = \frac{N}{2} \cdot (\frac{b \cdot N_b \cdot C_p}{f_c} + \frac{C_m}{f_m} + \frac{C_c}{f_c}) \quad cond2$$

where *cond1* and *cond2* represent CPU-bound case and memory-bound case, respectively. That is,

$$cond1 \quad represents \quad \frac{C_m}{f_m^0} \le \frac{b \cdot N_b \cdot C_p}{f_c^0} + \frac{C_c}{f_c^0}$$

$$cond2 \quad represents \quad \frac{C_m}{f_m^0} > \frac{b \cdot N_b \cdot C_p}{f_c^0} + \frac{C_c}{f_c^0}$$

where f_c^0 and f_m^0 represent initial CPU and memory frequencies, respectively.

The optimal performance objective is:

$$\min T_{total} = \begin{cases} \min(N \cdot (\frac{b \cdot N_b \cdot C_p}{f_c} + \frac{C_c}{f_c})) & cond1 \\ \min(\frac{N}{2} \cdot (\frac{b \cdot N_b \cdot C_p}{f_c} + \frac{C_m}{f_m} + \frac{C_c}{f_c})) & cond2 \end{cases} \quad (4)$$

3.3 Energy-Constrained Optimization Problem

Based on previous analysis, when CPU-bound case, energy-constrained optimization problem is represented as follows.

CPU-bound case

$$\min(N \cdot (\frac{b \cdot N_b \cdot C_p}{f_c} + \frac{C_c}{f_c})) \quad (5)$$

$$E_b \cdot b \cdot N_b \cdot N + k_1 \cdot f_c^2 \cdot C_c \cdot N + k_2 \cdot f_m^2 \cdot C_m \cdot N \le E_{budget} \quad (6)$$

$$C_m / f_m - (b \cdot N_b \cdot C_p + C_c) / f_c \le 0 \quad (7)$$

$$f_c^{'} \le f_c \le f_c^{''} \quad (8)$$

$$f_m^{'} \le f_m \le f_m^{''} \quad (9)$$

memory-bound case

$$\min(\frac{N}{2} \cdot (\frac{b \cdot N_b \cdot C_p}{f_c} + \frac{C_m}{f_m} + \frac{C_c}{f_c})) \quad (10)$$

$$E_b \cdot b \cdot N_b \cdot N + k_1 \cdot f_c^2 \cdot C_c \cdot N + k_2 \cdot f_m^2 \cdot C_m \cdot N \le E_{budget} \quad (11)$$

$$C_m / f_m - (b \cdot N_b \cdot C_p + C_c) / f_c > 0 \quad (12)$$

$$f_c^{'} \le f_c \le f_c^{''} \quad (13)$$

$$f_m^{'} \le f_m \le f_m^{''} \quad (14)$$

Here formula (7) and (12) guarantee CPU-bound case is still CPU-bound during frequency scaling and memory-bound case is still memory-bound. Inequality (8)-(9) and inequality (13)-(14) define the ranges of CPU and memory frequency scaling. The detailed meaning of each parameter and variable can be found in Table 1.

4 Compilation Strategy

Our energy optimization approach is profile-driven. The prototype implementation consists of four phases. It starts by instrumenting the original program at selected program locations (instrumentation phase). The instrumented code is then simulated, drawing out some profiled data, such as C_c, C_m (profiling phase). Once the profiling is done, all the parameters of this optimization problem are determined. The next work is

to obtain the optimal CPU and memory frequencies through solving the above optimization problem (solution phase). Finally, the frequency-setting calls are inserted at the appropriate situations so that the selected region is executed at the appropriate frequency and the rest of the program is executed at the highest frequency (code generation phase). In the future work, we will extend our approach to deal with other program segments besides regular loop segments. The whole compilation framework is shown in Figure 2.

In the next experiments, we will analyze the impact of parameters on our optimization approach and then choose a set of array-dominated applications to validate our optimization approach.

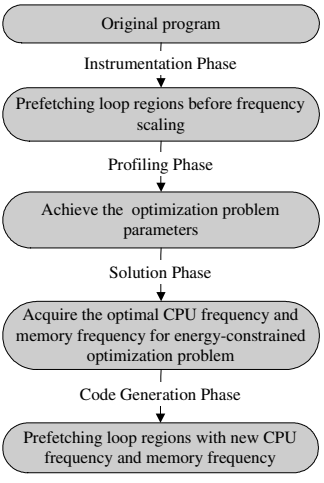


Fig. 2. Compilation Framework

Table 1. System parameters and program parameters for our optimal problemsolution

System Parameter	Meaning	Value
E_b	Energy consumption by a cache block prefetched	10 pJ
b	The number of cache block prefetched by a prefetching instruction	4
C_p	A prefetched cache block computation time	1 cycles
k_1	The coefficient for $P_c = k_1 \cdot f_c^3$	7.72e-27
k_2	The coefficient for $P_m = k_2 \cdot f_m^3$	1.09e-24
f_c^0	Initial CPU frequency	1000 MHz
f_m^0	Initial memory frequency	133 MHz
$f_c^{'}$	The lower bound of continous CPU frequency	433 MHz
$f_c^{''}$	The upper bound of continuous CPU frequency	1000 MHz
$f_m^{'}$	The lower bound of continuous memory frequency	83 MHz
$f_m^{''}$	The upper bound of continuous memory frequency	133 MHz
Program Parmeter	**Meaning**	**Value**
N	Prefetching loop iteration counts	Program specified
C_c	Computation time in once iteration (cycles)	Profiled
C_m	Memory access time in once iteration (cycles)	Profiled
N_b	The number of prefetching instructions in one iteration	Optimization specified
E_{budget}	Energy budget	specified

5 Experiments

5.1 Simulation Platform

In our simulation, CPU and memory both are DVS-enabled. We build a simple energy model for this DVS-enabled CPU and memory in terms of formula (2). That is,

$$P_c = (7.72e-27) \cdot f_c^3 \tag{15}$$

$$P_m = (1.09e-24) \cdot f_m^3 \tag{16}$$

The above formula (15) and (16) also refer to Transmeta's Crusoe TM5900 processor parameters [13]. The summary of power specifications for TM5900 CPU and

DDR can be found in [15]. All the parameters and profiled data for this optimal problem are shown in Table 1.

We use SimpleScalar tool set [14] to profile some necessary parameters such as C_c and C_m . Modified SimpleScalar tool set models a 1 GHz 4-way issue dynamically-scheduled processor. This simulator models all aspects of the processor including the instruction fetch unit, the branch predictor, register renaming, the functional unit pipelines, and the reorder buffer. This modified SimpleScalar tool set enables software prefetching through adding a prefetch instruction to the ISA of the processor model. In addition, our simulator also models the memory system in detail. A split 8-Kbyte direct-mapped L1 cache with 32-byte cache blocks, and a unified 256-Kbyte 4-way set-associative L2 cache with 64-byte cache blocks are assumed. Such cache configuration can meet our input data sets.

5.2 Parameters Analysis

5.2.1 The Impact of E_{budget} on Optimization

We only consider scarce-energy settings, where $E_{budget} < E_{bound}$. To describe the degree of energy scarcity, we define α as the ratio of energy budget to energy bound.

$$E_{budget} = \alpha \cdot E_{bound}$$

As α increases, energy budget is more approach to energy bound so that performance is more approach to the optimal value as Figure 3(a) and Figure 4(a) show, where blue solid lines represent execution time under different α values and black dot lines represent the optimal performance when energy bound is reached.

For CPU-bound case (Figure 3(a)), when $\alpha = 91.1\%$, performance has reached the optimal value instead of $\alpha = 100\%$. In contrast, for memory-bound case (Figure 4(a)), performance doesn't achieve the optimal level until $\alpha = 100\%$. This shows that in CPU-bound case, the optimal performance can be reached at less energy budget (less than energy bound). While in memory-bound case, the optimal performance must be reached with energy bound. In other words, in CPU-bound case, some energy can be saved without performance loss. While in memory-bound case, energy saving must cause performance loss.

Figure 3(b)-(c) and Figure 4(b)-(c) show CPU and memory frequency variances under the same conditions. In CPU-bound case (Figure 3(b)-(c)), when $\alpha > 91.1\%$ CPU frequency keeps at 1GHz while memory frequency is climbing and it reaches the highest point until $\alpha = 100\%$. Due to objective (5), the performance value in CPU-bound is determined by CPU frequency. Therefore, when CPU frequency reaches the highest value, performance reaches the optimal value. After that, memory frequency increase consumes unnecessary energy, which explains why some energy savings can be obtained without performance loss in CPU-bound case.

In contrast, in memory-bound case, performance is determined by both CPU frequency and memory frequency in terms of objective (10). Only when both of them reach the highest values, the performance reaches the optimal.

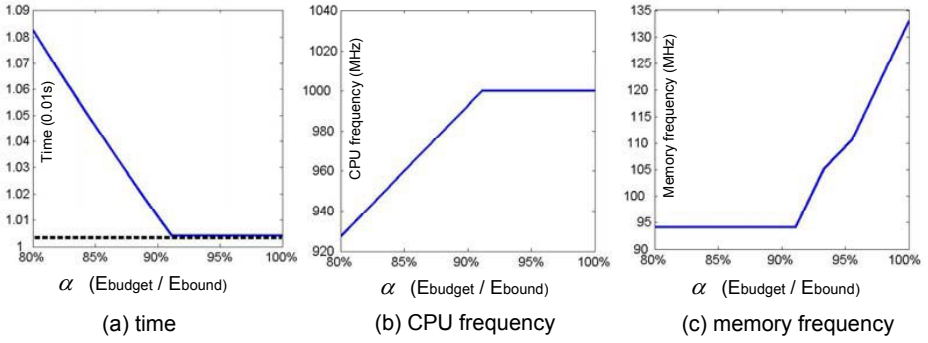


Fig. 3. The impact of Ebudget on the optimization results when CPU-bound case. Cc=1000, Cm=90.

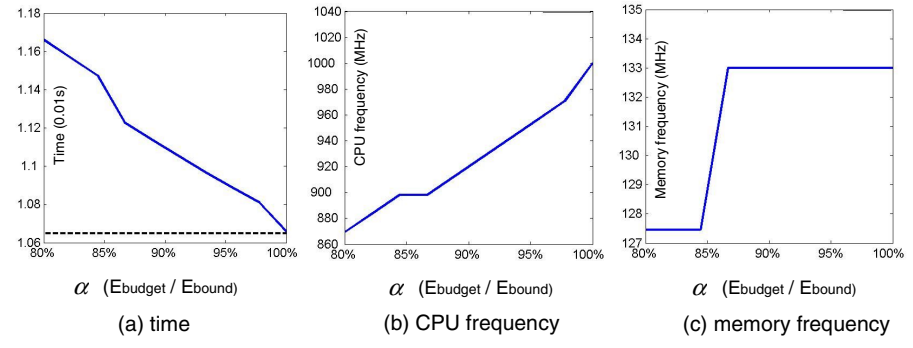


Fig. 4. The impact of Ebudget on the optimization results when memory-bound case. Cc=1000, Cm=150.

5.2.2 The Impact of C_c and C_m on Optimization

We also simulated the changes of time as C_c and C_m vary shown in Figure 5, where $\alpha = 90\%$. From point **Ai** to **Bi** (i=1, 2, 3, 4), the varying trends occur an exception. That is because the cases before **Ai** belong to memory-bound while the cases after **Bi** belong to CPU-bound and the calculation formulas for these two cases are different.

Figure 6 and Figure 7 give the optimal CPU and memory frequency settings as C_c and C_m vary. From these two graphs, the optimal CPU frequency value is non-decreasing as C_c increases while the optimal memory frequency value is non-increasing as C_c increases under a fixed C_m value. The C_c value and C_m value at points **Ai** and **Bi** (i=1, 2, 3, 4) are the same with those in Figure 5. Therefore, we can contrast Figure 6 and Figure 7 with Figure 5. Here $\alpha = 90\%$.

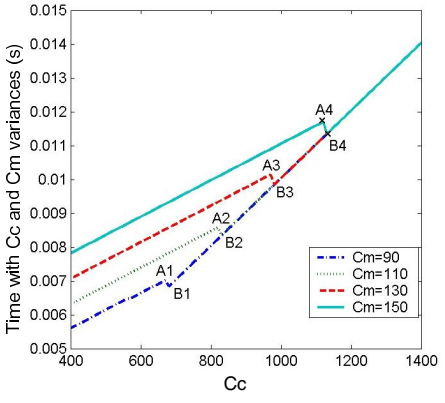


Fig. 5. Time variances with Cc and Cm variances

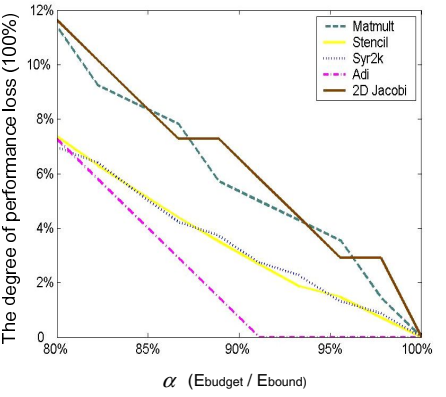


Fig. 8. The degree of performance loss for eachbenchmark

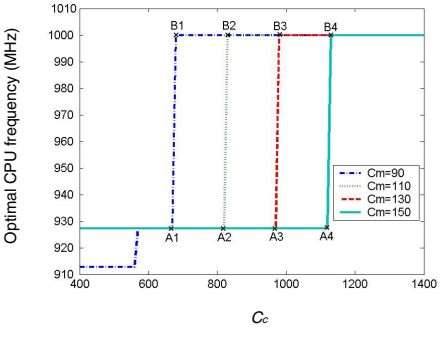


Fig. 6. Optimal CPU frequency variances with *Cc* and*Cm* variances

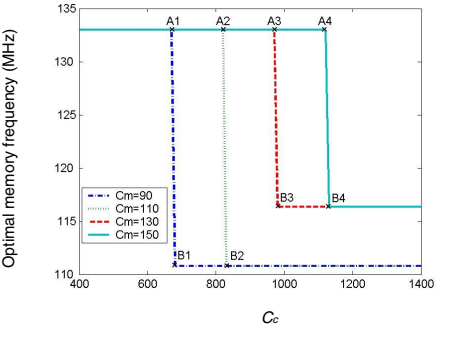


Fig. 7. Optimal memory frequency variances with *Cc*and *Cm* variances

5.3 Experimental Results

After the detailed parameters analysis, we choose a set of array-dominated applications to test the effectiveness of our energy optimization approach. They include Matmult, Stencil, Adi, 2D Jacobi and Syr2k. These benchmarks descriptions are given in Table 2.

Assume the time under energy constraint is *Time* and the time without energy constraint is *InitialTime*, the degree of performance loss can be calculated by formula (17).

$$Performance\ Loss\ (100\%) = \frac{\frac{1}{InitialTime} - \frac{1}{Time}}{\frac{1}{InitialTime}} \times 100\% \tag{17}$$

Table 2. The description of benchmarks

Benchmarks	description	The number of Array	Size
Matmult	matrices product	3	1024*1024
Adi	the core base benchmark of Livermore	6	1024*1024*3
2D Jacobi	2D Jacobi relaxation	2	1024*1024
Stencil	a stencil computing program for five dots	2	1024*1024
Syr2k	Rank-2K update computation program for solving zonal symmetry matrix from BLAS	3	1024*1024

Table 3. Optimal CPU and memory frequency settings and performance loss for each benchmark with variable

memory-bound benchmarks	α	CPU Fre. (MHz)	Memory Fre.(MHz)	Perf. Loss
Matmult	82.2%	898	122	9.3%
	86.7%	927	122	7.8%
	91.1%	942	127	5.0%
	95.6%	971	127	3.6%
	100%	1000	133	0
Stencil	82.2%	782	133	6.3%
	86.7%	840	133	4.4%
	91.1%	898	133	2.7%
	95.6%	942	133	1.5%
	100%	1000	133	0
Syr2k	82.2%	811	133	6.4%
	86.7%	869	133	4.2%
	91.1%	913	133	2.7%
	95.6%	956	133	1.3%
	100%	1000	133	0

memory-bound benchmarks	α	CPU Fre. (MHz)	Memory Fre.(MHz)	Perf. Loss
Adi	82.2%	942	88.6	5.8%
	86.7%	971	88.6	2.9%
	91.1%	1000	88.6	0
	95.6%	1000	111	0
	100%	1000	133	0
2D Jacobi	82.2%	898	122	10.2%
	86.7%	927	122	7.3%
	91.1%	942	127	5.8%
	95.6%	971	127	2.9%
	100%	1000	133	0

As α value varies, the performance loss for each benchmark is shown in Figure 8. Table 3 gives the detailed experimental data for the optimal CPU and memory frequency setting and performance loss percentage. In these five benchmarks, Matmult, Stencil and Syr2k belong to memory-bound cases and Adi and 2D Jacobi belong to CPU-bound cases. Experimental results validate our conclusions from the following aspects:

- Benchmark Adi reaches the optimal performance with 91.1% of energy bound, which validates that in CPU-bound case, the optimal performance can be reached under less energy budget (less than energy bound).
- For each benchmark belonging to memory-bound case, the optimal performance is only achieved with energy bound.
- The optimal CPU and memory frequency variances conform to Figure 3(b)-(c) and Figure 4(b)-(c).

6 Conclusions

Energy consumption is more and more important for battery-powered embedded systems due to the need for longer battery life and portability. Compiler-directed

optimization techniques are more and more concerned. In our article, we provide an energy-constrained prefetching optimization approach, which obtains the optimal performance with energy constraint through adjusting CPU and memory frequencies. For CPU-bound case and memory-bound case, frequency scaling shows the different characteristics. It is necessary to adjust both of CPU and memory frequencies for the optimization problem. We analyze the impact of several parameters on our energy-constrained optimization problem and draw some conclusions. Finally, experimental results validate the effectiveness of our approach.

Reference

1. Gang Qu. What is the Limit of Energy Saving by Dynamic Voltage Scaling? In ICCAD. page 560-563. 2001.
2. H. Saputra, M. Kandemir, N. Vijaykrishnan, M. J. Irwin, J. S. Hu, C-H. Hsu, U. Kremer. Energy-Conscious Compilation Based on Voltage Scaling. In LCTES'02-SCOPES'02, June 19-21, 2002, Berlin, Germany.
3. Fen Xie, Margaret Martonosi and Sharad Malik. Compile-Time Dynamic Voltage Scaling Settings: Opportunities and Limits. In PLDI'03, June 9-11, 2003, San Diego, California, USA.
4. Woo-Cheol Kwon and Taewhan Kim. Optimal Voltage Allocation Techniques for Dynamically Variable Voltage Processors. In DAC'03, June 2-6, 2003, Anaheim, California, USA.
5. Xiaobo Fan, Carla S. Ellis and Alvin R. Lebeck. The Synergy between Power-aware Memory Systems and Processor Voltage Scaling. In *Proceedings of the Workshop on Power-Aware Computer Systems (PACS'03)*, Dec 2003.
6. Todd C. Mowry. Tolerating Latency Through Software-Controlled Data Prefetching. Ph.D. thesis, Stanford University, Computer System Laboratory, March 1994.
7. Shimin Chen, Phillip B. Gibbons and Todd C. Mowry. Improving Index Performance through Prefetching. In *Proceedings of the 2001 SIGMOD International Conference on Management of Data*. Page 235-246. May 2001.
8. Abdel-Hameed Badawy, Aneesh Aggarwal, Donald Yeung, and Chau-Wen Tseng. The Efficacy of Software Prefetching and Locality Optimizations on Future Memory Systems. Journal of Instruction-Level Parallelism. June 2004.
9. Ricardo Bianchini and Beng-Hong Lim. Evaluating the Performance of Multithreading and Prefetching in Multiprocessors. Journal of Parallel and Distributed Computing (JPDC), special issue on Multithreading for Multiprocessors, August 1996.
10. C. Hsu, U. Kremer, and M. Hsiao. Compiler-Directed Dynamic Voltage/Frequency Scheduling for Energy Reduction in Microprocessors. In *International Symp. On Low Power Electronics and Design (ISLPED)*, pages 275-278, August 2001.
11. Chung-Hsing Hsu. Compiler-Directed Dynamic Voltage and Frequency Scaling for CPU Power and Energy Reduction. Ph. D. Dissertation. New Brunswick, New Jersey. October 2003.
12. J. Pouwelse, K. Langendoen, and H. Sips. Dynamic Voltage Scaling on a Low-Power Microprocessor. In *the Seventh Annual International Conference on Mobile Computing and Networking 2001*, pages 251-259, 2001.
13. Crusoe Processor Model TM5700/TM5900 Data Book.http://www.transmeta.com/crusoe_docs/tm5900_databook_040204.pdf

14. D. Burger and T. M. Austin. The SimpleScalar Tool Set, Version 2.0. CS TR 1342, University of Wisconsin-Madison, June 1997.
15. Juan Chen, Yong Dong, Xue-jun Yang. Energy Optimization for Software Prefetching in Embedded Applications. *In the proceedings of Asia and South Pacific International Conference on Embedded SoCs (ASPICES 2005)*. July 5-8, 2005, Bangalore, India.

An Energy Reduction Scheduling Mechanism for a High-Performance SoC Architecture

Slo-Li Chu

Department of Information and Computer Engineering,
Chung Yuan Christian University, Chung-Li, Taiwan, R.O.C.
slchu@cycu.edu.tw

Abstract. Continuous improvements in semiconductor technology are supporting new classes of System-on-a-Chip (SoC) architectures that combine extensive processing logic with high-density memory. Such architectures are generally called Processor-in-Memory (PIM) or Intelligent Memory (I-RAM) and can support high-performance computing by reducing the performance gap between the processor and the memory. The PIM architecture combines various processors in a single chip. These processors are characterized by their computation, memory-access and power consumption capabilities. Therefore, a novel parallelizing system, SAGE II, has been developed to identify their capabilities and dispatch the most appropriate jobs to them in order to exploit the advantages of PIM architectures. However, the SAGE II system only can deal with performance issues but power consumption is gradually becoming an important issue of current computing systems. This paper provides a new low-power transformation mechanism, called Energy-Oriented Power Reduction Scheduling (EOPRS), to extend the capability of SAGE II system. It can reduce the power consumption for the Processor-in-Memory system without losing execution performance. The detailed EOPRS transformation technique is presented later. The experimental results of several benchmarks are also discussed.

Keywords: EOPRS, SAGE II, SoC, Processor-in-Memory, Power Reduction.

1 Introduction

In current high-performance computer architectures, processors run many times faster than the computer's main memory. This performance gap is often referred to as the Memory Wall [2]. This gap can be reduced using the System-on-a-Chip (SoC) strategy, which integrates the processor(s) and memory on a single chip. Accordingly, many researchers have considered integrating computing logic and high density DRAM on a single die [3, 5, 8, 10] as so-called Processor-in-Memory (PIM) or Intelligent RAM (IRAM). This class of architectures constitutes a hierarchical hybrid multiprocessor environment that involves host (main) and memory processors. The host processor (P.Host) is more powerful but have a deep cache hierarchy and higher latency when accessing memory. In contrast, memory processors (P.Mem) are typically less powerful but have a lower latency in memory access. With respect to energy consumption, host processor consumes more energy than memory processors when computing and accessing memory. These capability differences make parallelization and energy reduction more difficult.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 281 – 290, 2005.
© IFIP International Federation for Information Processing 2005

The main problems addressed are how to dispatch appropriate tasks to these various processors according to their characteristics, to reduce their execution time, energy consumption, or both, and how to partition the original program so that it can be executed on these heterogeneous processor mixtures. Accordingly, an automatic source-to-source parallelizing and optimizing system, SAGE II (Statement- Analysis-Grouping- Evaluation II) is designed and implemented. The main difference between SAGE II and other parallelizing systems is that it uses statement rather than iteration as the basic unit of analysis. By integrating statement splitting, weight evaluation and several scheduling mechanisms, SAGE II can automatically analyze Fortran source program; partition it into statements; generate a Weight Partition Dependence Graph (WPG), determine the weight of each block in the WPG, and schedule these blocks to improve the performance by dispatching the most suitable blocks for execution on host and memory processors. To extend the capability of reducing power consumption of software running on the SoC architecture, this study presents a novel scheduling technique, called Energy-Oriented Power Reduction Scheduling (EOPRS), for to reduce the energy consumed for the one-P.Host and one-P.Mem configuration without losing performance. The remainder of this paper is organized as follows: Section 2 introduces the related work. Section 3 describes the PIM architecture we adopt. Section 4 describes SAGE II system. Section 5 provides the EOPRS mechanism. The experimental results are shown in Section 6. Finally, conclusion is given in Section 7.

2 Related Work

Since energy limited the growth of the mobile computers and embedded computing devices, it is very important to reduce the consumption of energy and increase battery lifetimes. Therefore the analysis and optimization of the energy consumed by all components become important not only for battery-powered personal devices, but also for large computing systems.

Current work on energy saving using software can be classified into two categories. In one, special architecture is required to support the software. For example, an additional mini cache between the I-Cache and the CPU core called the "L-Cache" or "I-Filter" can be used [1, 5, 6]. Such mini cache buffers only the reused instructions within loops. It consumes less energy than I-Cache and so eliminates the fetching of unnecessary instructions fetching, reduce signal switching activity and dissipated energy. It requires compilers to cooperate to the rearrange original codes to ensure that it fits the L-Cache. However, it also has the disadvantage of requiring a special architecture.

Another category of work considers instruction-level scheduling mechanisms to reduce energy consumption [4, 7, 9, 12]. If a previously executed instruction differs from the present instruction, the control circuit must activate different hardware functional units. Switching consumes energy, so appropriately reordering instructions in a program reduce the energy consumption. This method need to measure the inter-instruction effects and establishes an energy transition cost table, before performing instruction-level scheduling according to the cost of switching. However, such a

method has two disadvantages. First, the source program must be written in a low-level language such that the scheduler cannot understand all of the behavior of this program, limiting the application of high-level transformation techniques, such as tiling. Second, the cost of switching instructions is only around 2% ~ 10% of the base cost of the instructions; therefore the energy reduction is not significant.

3 The Processor-in-Memory Architecture

Figure 1 depicts the organization of the PIM architecture evaluated in this study. It contains an off-the-shelf processor, P.Host, and a PIM chip. The PIM chip integrates one memory processor, P.Mem, with 64 Mbytes of DRAM. This architecture is derived from FlexRAM [5]. The tiny memory processors (P.Array) in the original FlexRAM are omitted to reduce the complexity of analysis. The techniques presented in this paper involve a one P.Host and one P.Mem configuration, and can be extended to support multiple P.Mems.

Table 1. Major parameters of the PIM architecture

P.Host	P.Mem	Bus & Memory
Working Freq: 800 MHz	Working Freq: 400 MHz	Bus Freq: 100 MHz
Dynamic issue Width: 6	Static issue Width: 2	P.Host Mem RT: 262. 5 ns
Integer unit num: 6	Integer unit num: 2	P.Mem Mem RT: 50. 5 ns
Floating unit num: 4	Floating unit num: 2	Bus Width: 16 B
FLC_Type: WT	FLC_Type: WT	Mem_Data_Transfer: 16
FLC_Size: 32 KB	FLC_Size: 16 KB	Mem_Row_Width: 4K
FLC_Line: 64 B	FLC_Line: 32 B	
FLC_Replace policy: LRU	FLC_Replace policy: LRU	
SLC_Type: WB	SLC: N/A	
SLC_Size: 256 KB		
SLC_Line: 64 B		
Replace policy: LRU		
Branch penalty: 4	Branch penalty: 2	
P.Host_Mem_Delay: 88	P.Mem_Mem_Delay: 17	

** FLC stands for the first level cache, SLC for the second level cache, BR for branch, RT for round-trip latency from the processor to the memory, and RB for row buffer.*

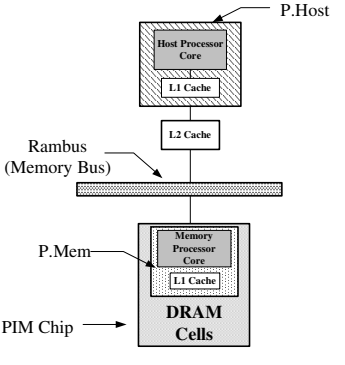


Fig. 1. Organization of the PIM architecture

Table 2. Power consumption parameters of the PIM architecture

	P.Host		P.Mem	
Integer Unit	ALU	194.46	ALU	97.23
	Shift	194.46	Shift	97.23
	Mult/div	210.45	Mult/div	105.23
	Branch	194.46	Branch	97.23
Load/Store Unit	Load	194.46	Load	97.23
	Store	194.46	Store	97.23
Floating point Unit	Add/Sub	210.45	Add/Sub	105.23
	Mult	210.45	Mult	105.23
	Div	210.45	Div	105.23
Clock energy	957		26.24	

Table 1 and Table 2 list the main architectural parameters of the PIM architecture. P.Host is a six-issue superscalar processor that allows out-of-order execution and runs at 800MHz, while P.Mem is a two-issue superscalar processor with in-order capability and runs at 400MHz. There is a two-level cache in P.Host and a one-level cache in P.Mem. P.Mem has lower memory access latency than P.Host since the former is integrated with DRAM. Thus, computation-bound codes are more suitable for running on the P.Host, while memory-bound codes are preferably running on the P.Mem to increase efficiency.

The PIM chip is designed to replace regular DRAMs in current computer systems, and must therefore conform to a memory standard that involves additional power and ground signals to support on-chip processing. One such standard is Rambus, so the PIM chip is designed with a Rambus-compatible interface.

4 The SAGE II System

Most current parallelizing compilers focus on transforming loops to execute iterations concurrently, in a so-called iteration-based approach. This approach is suited to homogeneous and tightly coupled multi-processor systems. However, it has an obvious disadvantage for heterogeneous multi-processor platforms because iterations have similar behavior but the capabilities of heterogeneous processors are diverse. Therefore, a different approach is adopted here, using the statements in a loop as a basic analysis unit, called statement-based approach, to develop the SAGE system.

SAGE II (Statement-Analysis-Grouping-Evaluation II) is an automatic parallelizing compiler, which partitions and schedules an original program to exploit the specialties of the host and the memory processor. At first, the source program is split into blocks of statements according to dependence relations. Then, the Weighted Partition Dependence Graph (WPG) is generated, and the weight of each block is evaluated. Finally, the blocks are dispatched to either the host or the memory processors, according to which processor is more suitable for executing the block. The major difference between SAGE II and other parallelizing systems is that it uses a statement rather than an iteration as the basic unit of analysis. This approach can fully exploit the characteristics of statements in a program and dispatch the most suitable tasks to the host and the memory processors.

Table 3 presents a simple example to demonstrate the advantages of statement-based parallelization. The program is fully parallelizable and can be partitioned into statements or iterations. The table lists the assumed statement weights for the P.Host and P.Mem. Table 4 shows five parallelization cases and their execution times. The first two involve executing the program solely on P.Host and P.Mem, respectively. Case 3 parallelizes the program using conventional parallelizing compilers, such as SUIF [2] or Polaris [2] to identify the parallelizable loops and dispatch them for execution on P.Host and P.Mem. This approach only achieves good speedup for processors with homogeneous capabilities (including memory access latency, computing power, and so on). In case 4, the iterations are dispatched to the processors according to the processors' capabilities, but the compiler does not consider the discrepancies among processors in executing statements. Case 5 uses the statement-based analysis approach (i.e., optimized by SAGE

II). This approach outperforms all the others since it dispatches statements to P.Host and P.Mem by accounting for the characteristics of statements and the capabilities of processors, motivating the development of the SAGE II system for asymmetric multiprocessor environments. The compiling sequence of the SAGE II is shown in Figure 2. The major stages are described later.

Table 3. A fully parallelizable program

Program	*Weight for *PH*	*Weight for *PM*
DO I = 1 to N		
S1: A= A mod B	3	6
S2: C= D[I]+E	5	1
S3: F= G[I]+H[I]	6	2
ENDDO		

**PH* stands for host processor and *PM* for memory processor.

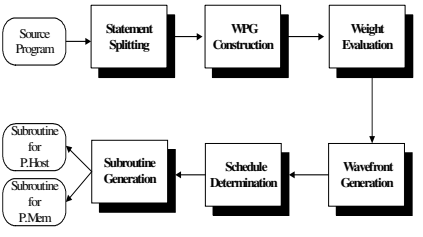


Fig. 2. The sequence of compiling stages in SAGE II

Table 4. Five parallelizing cases and their execution times

	Description	*Execution Time*
1	Host processor operates solely	Latency = [PH(S1)+PH(S2)+PH(S3)]*# of iterations = (3+5+6)N = 14 N
2	Memory processor operates solely	Latency =[PM(S1)+PM(S2)+PM(S3)]* # of iterations = (6+1+2)N = 9 N
3	Host and memory processors cooperate in symmetric workload	Latency = max((3+5+6)* 0.5N, (6+2+1)* 0.5 N) = 7 N
4	Host and memory processors cooperate in asymmetric workload by parallelizing the *iteration* space of the loop	The workload is dispatched based on the capability ratio of PH and PM obtained from Case 1 and Case2, i.e., PH: PM = 9:14. Hence, latency=14* (9/(9+14))N= 5.48 N
5	Host and memory processors cooperate in asymmetric workload by SAGE optimization	Latency = max (PH(S1) * N, PM(S2,S3)*N) = 3 N (Here S1 is more suitable for P.Host, but S2 and S3 are more suitable for P.Mem.)

4.1 Statement Splitting and WPG Construction [2]

Statement Splitting splits the dependence graph by Node Partitioning as introduced in [2]. WPG Construction constructs the Weighted Partition Dependence Graph (WPG), to be used in the subsequent stages of Weight Evaluation, Wavefront Generation and Schedule Determination.

The definitions relevant to Statement Splitting are introduced in the literature [2]. It is ignored for the page limitation but still listed the Statement Splitting algorithm (Algorithm 1) to partition the loops:

Algorithm 1. (Statement Splitting Algorithm)

Given a loop L = (i_1, i_2, …. i_n) (s_1, s_2, …. s_d)

Step 1: Construct dependence Graph G by analyzing subscript expressions and index pattern.

Step 2: Establish a node partition Π on G as defined in Definition 2. If there are large blocks caused by control dependence relations, convert control dependence into data dependence first [2], and then partition the dependence graph.

Step 3: On the partition Π, establish a weighted partition dependence graph WPG(B,E) defined in Definition 3.

4.2 Weight Evaluation

Two approaches to evaluating weight can be taken. One is to predict the execution time of programs by profiling the dominant parts [2]. The other considers the operations in a statement and estimates the program execution time by looking up an operation weight table [2]. The former method called code profiling may be more accurate, but the predicted result cannot be reused; the latter called code analysis can determine statements for suitable processors but the estimated program execution time is not sufficiently accurate. Hence, the Self-Patch Weight Evaluation scheme was designed to combine the benefits of both approaches. It integrates these two approaches together by analyzing code and searching weight table first to estimate the weight of a block. If the block contains unknown operations, the patch (profiling) mechanism is then activated to evaluate the weights of unknown operations. The obtained operation weights are added into the weight table for next look-up. For a detailed description of this scheme, please refer to [2]

5 Energy-Oriented Power Reduction Scheduling Mechanism

5.1 Algorithm Description

This paper has proposed an automatic source-to-source parallelizing compiler, SAGE II (Statement- Analysis- Grouping- Evaluation), which integrates statement splitting, weight evaluation, scheduling and optimizing mechanisms. However, the best execution schedule in terms of performance is not usually the same as the best execution schedule in terms of energy. This section focuses on reducing the energy consumption of PIM architecture. For the reason mentioned in the previous descriptions, statements are adopted as the analyzing units. Each statement unit has a delay weights and an energy weight for P.Host and P.Mem, respectively, and will be assigned to the most suitable processor according to these weights. Here we introduce

---

Algorithm 2: Energy-Oriented Power Reduction Scheduling

[Input]

WPG=(P, E): the order of blocks and weights are determined.

Percent: the percent user want to constrain.

[Output]

An execution wavefront schedule $WF= \{Wf_1, Wf_2, ...\}$ where $Wf_1=\{PH(b_a...b_b), PM(b_c...b_d)\}$ in which $PH(b_a...b_b)$ means that blocks $b_a...b_b$ will be assigned to PHost in wavefront *i*, $PM(b_c...b_d)$ means that blocks $b_c...b_d$ will be assigned to PMem in wavefront *i*.

[Intermediate]

W: a working set of blocks to be visited.

max_pred_O(b_i): the maximum execution order for all b_i's predecessor blocks.

min_pred_O(b_i): the minimum execution order for all b_i's predecessor blocks.

wf_tmp: working sets of blocks for wavefront scheduling.

ph_tmp(h), pm_tmp(m): working arrays to store the blocks for wavefront scheduling.

ph_d_wei(h), pm_d_wei(m) working arrays to store the total delay weights of blocks .

ph_e_wei(h), pm_e_wei(m) working arrays to store the total energy weights of blocks .

max_wf: the maximum number of wavefront.

PHEW_origin(max_wf): the energy weight of b_i for Phost each wavefront.

Deadline(max_wf): the deadline of each wavefront.

PHDW(b_i): the delay weight of b_i for PHost.

PMDW(b_i): the delay weight of b_i for PMem.
PHEW(b_i): the energy weight of b_i for PHost.
PMEW(b_i): the energy weight of b_i for PMem.

[Method]

```
for each  b_i ∈P do /*Initialization and weight determination for each blocks */
    O_i = 0; evaluate PHDW(b_i), PMDW(b_i), PHEW(b_i) and PMEW(b_i) by weight evaluation;
end for
W = P;
for each b_i with no predecessors do /* Execution order assignment */
    O_i = 1; W = W – { b_i };
end for
done = False; max_wf=0;
while done = False AND W ≠ ϕ do
    done = True
    for each  b_i ∈ W do
        if min_pred_O(b_i) = 0  then done = False
        else
            O_i = max_pred_O(b_i)+1; W=W-{ b_i }; max_wf = max(max_wf, O_i);
        end if
    end for
end while
energy_temp = 0
for j = 1 to max_wf
    wf_tmp=ph_tmp=pm_tmp={ϕ};h=m=Opt_energy(j)=PHEW_origin(j)=Deadline(j)=0;
    ph_d_wei(j)=ph_e_wei(j)=pm_d_wei(j)=pm_e_wei(j)=0;
    store all of b_i whose O_i = j  in wf_tmp;
    for each  b_i∈ wf_tmp
        if  PHEW(b_i ) - PMEW(b_i) ≤0
            h = h + 1; Store b_i in ph_tmp (h);
            ph_d_wei(j)=ph_d_wei(j)+PHDW(b_i ); ph_e_wei(j)=ph_e_wei(j)+PHEW(b_i );
        else
            m = m + 1; Store b_i in pm_tmp (m);
            pm_d_wei(j)=pm_d_wei(j)+PMDW(b_i ); pm_e_wei(j)=pm_e_wei(j +PMEW(b_i );
        end if
        PHEW_origin(j)=PHEW_origin(j)+PHEW(b_i );
        Opt_energy(j) = Opt_energy(j) + Min(PHEW(b_i ), PMEW(b_i ));
        Sort ph_tmp (h) in decreasing order by PHDW(b_i );
        Sort pm_tmp (m) in decreasing order by PMDW(b_i);
    end for
    Deadline(j)=Opt_energy(j)+(PHEW_origin(j)–Opt_energy(j))*(1–Percent);
    Deadline(j)=Deadline(j)+energy_temp; p = q = 0; done = FALSE;
    if  ph_d_wei(j) ≥ pm_d_wei(j)
        for p =1 to h
            ph_e_wei(j) = ph_e_wei(j) – PHEW ( ph_tmp(p))
            pm_e_wei(j) = pm_e_wei(j) + PMEW ( ph_tmp(p))
            ph_d_wei(j) = ph_d_wei(j) – PHDW ( ph_tmp(p))
            pm_d_wei(j) = pm_d_wei(j) + PMDW ( ph_tmp(p))
            if  ph_e_wei(j) + pm_e_wei(j) ≤ Deadline(j)
                ph_tmp = ph_tmp – {ph_tmp(p)}; pm_tmp = pm_tmp + {ph_tmp(p)};
            else
                pm_d_wei(j) = pm_d_wei(j) – PMDW ( ph_tmp(p))
                ph_d_wei(j) = ph_d_wei(j) + PHDW ( ph_tmp(p))
                pm_e_wei(j) = pm_e_wei(j) – PMEW ( ph_tmp(p))
                ph_e_wei(j) = ph_e_wei(j) + PHEW ( ph_tmp(p))
            end if
        end for
    else
        for q =1 to m
            ph_e_wei(j) = ph_e_wei(j) + PHEW ( pm_tmp(q))
            pm_e_wei(j) = pm_e_wei(j) – PMEW ( pm_tmp(q))
            ph_d_wei(j) = ph_d_wei(j) + PHDW ( pm_tmp(q))
            pm_d_wei(j) = pm_d_wei(j) – PMDW ( pm_tmp(q))
            if  ph_e_wei(j) + pm_e_wei(j) ≤ Deadline(j)
                pm_tmp = pm_tmp – {pm_tmp(q)}
                ph_tmp = ph_tmp + {pm_tmp(q)}
            else
                ph_d_wei(j) = ph_d_wei(j) – PHDW ( pm_tmp(q))
```

```
            ph_e_wei(j) = ph_e_wei(j) – PHEW ( pm_tmp(q))
            pm_d_wei(j) = pm_d_wei(j) + PMDW ( pm_tmp(q))
            pm_e_wei(j) = pm_e_wei(j) + PMEW ( pm_tmp(q))
          end if
        end for
    end if
    energy_temp=Deadline(j)–( ph_e_wei(j)+pm_e_wei(j)); Wfj ={ph_tmp, pm_tmp };
end for
```

the Energy-Oriented Power Reduction Scheduling, EOPRS, which allows the user to specify the energy limit. If the energy limit is met, the scheduler will use the performance-oriented principle to schedule the blocks. For simplicity, this paper considers only the system with a single P.Host and single P.Mem. The rest of this section presents the mechanisms in detail. Section 6 presents the experimental results.

This EOPRS approach enables the user to adjust the energy reduction ratio to get more speedup. In this approach, the energy weights and delay weights of each block in the WPG are determined first. Then the execution order for each block is determined according to their dependence relations. The blocks that can be executed simultaneously are assigned to a wavefront. Then the scheduler dispatches blocks in a single wavefront according to their energy weights, minimizing the energy consumption. According to the energy specified by the user, the maximum energy acceptable consumption, called the "deadline" is determined herein. The deadline is given by minimum energy consumption + (original energy consumption - minimum energy consumption) * (1 - energy reduced limit %). Then, the mechanism can automatically adjust schedule sequence to obtain the maximum potential speedup, such that the energy consumption is less than the maximum acceptable. The details mechanism of this approach is presented in Algorithm 2.

6 Experimental Results

This experiment focuses on the one-P.Host and one-P.Mem configuration of PIM architecture to evaluate the energy reduction scheduling mechanism mentioned in Section 5. The energy consumption parameters of this PIM architecture can be found in the Table 2.

The evaluated applications include five programs - swim from SPEC95, cgemm from BLAS3, ft and cg from the serial version of NAS, and mdg from Perfect Benchmark. Table 5 shows the results of the experiment. "Std-" denotes that the applications are executed on P.Host alone; "Opt-" denotes that the applications are transformed by the one-P.Host and one-P.Mem performance scheduling mechanism mentioned in literature [2] for execution on one P.Host and one P.Mem simultaneously; "Ene-" denotes that the applications are transformed by "Energy-Oriented Energy Reduction Scheduling" as mentioned in Section 5 with the energy reduced ratio set to 70%.

In Table 5, swim, cg, and cgemm exhibit good speedup "Opt-" case because they all can be partitioned into many blocks for scheduling to P.Host and P.Mem, according to the characteristics of blocks and processors. Also, they can achieve a good energy reduction ratio; ft can obtain good speedup "opt-" case but some performance in "Ene-" case is lost because its block sizes are very different.

Table 5. Experimental results of five benchmarks

benchmark	form	Execution cycles	Speedup	Energy consumption (joule)	Energy Reduce Ratio
swim	Std-	228289321	1.00	43.49	0
	Opt-	116669760	1.96	35.29	18.85%
	Ene-	262401518	0.87	28.18	35.20%
cg	Std-	91111840	1.00	19.58	0
	Opt-	51230772	1.78	14.73	24.77%
	Ene-	67993910	1.34	12.66	35.34%
cgemm	Std-	257528272	1.00	50.26	0
	Opt-	147769271	1.74	38.63	23.14%
	Ene-	238641808	1.07	33.02	34.30%
ft	Std-	544099032	1.00	111.9	0
	Opt-	368981380	1.47	86.59	22.62%
	Ene-	549370943	0.99	74.41	33.50%
mdg	Std-	174506369	1.00	65.5	0
	Opt-	160335813	1.08	63.95	2.37%
	Ene-	385931357	0.45	41.68	36.37%

Scheduling these blocks is difficult when the power reduction ratio is set to 70%. In contrast, mdg is intrinsically sequential. It can only be partitioned into several large blocks, preventing the generation of load-balance schedules, so the speedup of "Opt-" case is only 1.08. The EOPRS can further reduce the energy but the speedup is decreased to 0.45. Thus, the user can select different scheduling approaches according to their requirements.

7 Conclusions

In this paper, we presented a new program transformation technology, Energy-Oriented Power Reduction Scheduling, EOPRS. Integrated with SAGE II system, EOPRS reduces the energy consumption on the PIM System. This approach allows the user to set the speedup limit. If the speedup limit is met, then the scheduler will use the energy reduction principle to schedule the blocks. Results of the experiments demonstrate the capabilities of the extended SAGE II system. In the experiment, the energy consumption was reduced by up to 36.37%. The author believe that the techniques proposed here can be extended to run on DIVA, EXECUBE, FlexRAM, and other high-performance SoC architectures by slightly modifying the code generator of the SAGE system.

Acknowledgement

This work is supported in part by the National Science Council of Republic of China, Taiwan under Grant 94-2213-E-033-032-

References

[1] Bajwa, R. S., et al.: Instruction Buffering to Reduce Power in Processors for Signal Processing. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol.5(4). IEEE, (1997) 417-424.
[2] Chu, S. L., Huang, T. C.: SAGE: A Comprehensive Parallelizing Framework for Processor-in-Memory Architectures. Journal of Systems Architecture. Vol. 50 (1). Elsevier Science, (2004) 1-15.
[3] M. Hall, et al,: Mapping Irregular Applications to DIVA, a PIM-Based Data-Intensive Architecture. Proc. of 1999 Conference on Supercomputing. ACM, (1999).
[4] Horowitz, M., Indermaur, T., Gonzalez, R.: Low-Power Digital Design. Proc. of Symposium on Low Power Electronics. 8-11, (1994).
[5] Kang, Y., et al.: FlexRAM: Toward an Advanced Intelligent Memory System. Proc. of International Conference on Computer Design. (1999).
[6] Kin, J., et al.: The filter cache: an energy efficient memory structure. Proc. of Thirtieth Annual IEEE/ACM International Symposium on Microarchitecture. (1997) 184-193.
[7] Lee, L. H., Moyer, B., Arends, J.: Instruction Fetch Energy Reduction Using Loop Caches for Embedded Applications with Small Tight Loops. Proc. of 1999 International Symposium on Low Power Electronics and Design. (1999) 267-269.
[8] Oskin, M., Chong, F. T., Sherwood, T.: Active Page: A Computation Model for Intelligent Memory Computer Architecture. Proc. 25th Annual International Symposium on Computer Architecture. (1998) 192 –203.
[9] Parikh, A.., et al.: Energy-Aware Instruction Scheduling. Proc. of 7th International Conference on High Performance Computing-HiPC (2000) 335-344.
[10] Patterson, D., et al.: A Case for Intelligent DRAM. IEEE Micro, IEEE (1997) 33-44.
[11] Press,W. H., et al., Numerical Recipes in Fortran 77. Cambridge University Press (1992).
[12] Raghunathan, A., Jha, N. K., Dey, S.: High-Level Power Analysis and Optimization. Kluwer Academic Publishers, (1998).
[13] Tiwari, V., Malik, S., Wolfe, A.: Power Analysis of Embedded Software: a First Step Towards Software Power Minimization. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 2 (4). IEEE (1994) 437-445.

A New Buffer Planning Algorithm Based on Room Resizing*

Hongjie Bai, Sheqin Dong, Xianlong Hong, and Song Chen

Department of Computer Science and Technology,
Tsinghua University, Beijing, China
bhj04@mails.tsinghua.edu.cn, dongsq@mail.tsinghua.edu.cn

Abstract. This paper studies the buffer planning problem for interconnect centric floorplanning. Dead-spaces not held by blocks are the available location for buffer insertion. To make best use of these spaces for buffer requirements, we have to move blocks so that blocks' room size is adjusted and dead-spaces could be redistributed. In this paper, we introduce a new algorithm to move blocks not only within its room, but also in the space currently held by other blocks by pushing away these blocks if necessary without violating the topological and the total area. After applying this method of redistributing dead-spaces, the number of nets satisfying delay constraint can be optimized.

1 Introduction

As the VLSI circuits are scaled into nanometer dimensions, interconnect plays a dominant role in the performance and cost. To ensure the timing closure of design, interconnects must be considered as early as possible. Buffer insertion is among most effective ways to reduce interconnect delay which breaks long interconnects into shorter ones so that overall delay can be reduced. However, most buffer insertion algorithms were designed for post-layout interconnect optimization. Obviously it is not suitable if we have no global planning for tens of thousands of nets that need buffers to be inserted. Also, buffers take up silicon resources, so it is necessary to plan buffer as early as possible in the design of VLSI.

1.1 Previous Work

[3] introduced the concept of feasible region which employs Elmore delay model to generate buffer blocks. [5] extends it by adding independence to feasible region and also tries to relieve routing congestion. [6] proposed an optimal algorithm which assumes only one buffer per net. [7] use multi-commodity flow-based approach which allocates buffers to pre-existing buffer blocks. [8] proposes

* This paper is supported by National Nature Science Foundation of China (NSFC): 60473126, NSFC 60121120706 and National Natural Science Foundation of USA (NSF) CCR-0096383, Hi-Tech Research & Development (863) Program of China 2004AA1Z1050

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 291–299, 2005.
© IFIP International Federation for Information Processing 2005

a routability driven floorplanning. Most of these methods try to allocate buffers into dead spaces, however none of these methods take concern of using dead spaces effectively enough so that many dead spaces are left unused while still many buffers have no space to insert. [2] introduces a dead space redistribution-based method to make better use of dead space for buffer insertion. However, there exists a drawback that it can only move blocks within its room. Figure 1(a) is an example. With the algorithm introduced in [2], block C can't be moved horizontally so that some buffer requirements can't be satisfied. However, if E could move right a little to lend some space to C, all buffers can be successfully inserted as shown in Fig. 1(b) .

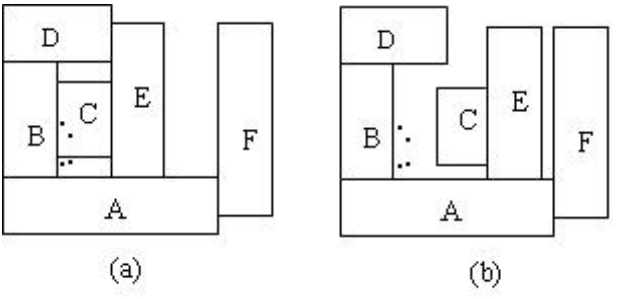


Fig. 1. Buffers can't be placed in (a) if C only moves within its room, but can be successfully placed in (b) if we enlarge C's room

1.2 Our Contribution

Except for moving blocks within their rooms, the rooms' size could be adjusted without changing the blocks' relative positions and buffer planning could be improved greatly. In that case, if we wish to find the optimal solution to insert buffers, we have to try moving blocks which may even push other blocks. So our algorithm is concentrated on moving blocks as much as possible in order to provide dead-spaces for buffer, at the same time, the dead-spaces of the total placement are managed systematically. This new method will be very effective for IP based SOC planning. For instance, for design reuse where topological relations among blocks could not be changed, this method can make blocks move in a range as much as possible to optimize buffer planning.

The rest of the paper is organized as follows, section 2 gives a problem definition and a brief overview to the background including Feasible Region and Corner Block List. Section 3 discusses our algorithm. Section 4 and section 5 give the experimental result and conclusion.

2 Preliminary

2.1 Problem Formulation

Given an initial placement and the nets with timing constraint, for the nets that does not satisfy timing closure, we determine the buffer number needed for each

net and the definite buffer locations. Since buffer requires silicon resources, it can only be located in dead-spaces which are not held by blocks. To best satisfy buffer requirements, blocks have to be moved to make more nets meet timing closure while the total area and topology remain unchanged.

2.2 Corner Block List (CBL)

Corner block list (CBL) is a topological representation introduced in [1]. CBL uses three lists S,L,T to represent a floorplan. Each member in S represents a block, L and T are lists of 0 and 1. While applying a CBL to a floorplan, we assign the blocks in S one by one to the up-right corner. The corresponding number in L denotes the direction. 0 makes the new block come from the top direction and 1 makes the new block come from the right direction. The segments lying on the left and bottom boundary of the corner block compose a T-junction which defines the orientation of the corner block. The number of consecutive 1 in T means the number of T-Junctions the new block will cover. The insertion process of a block is shown in Fig. 2. Details can be found in [1] .

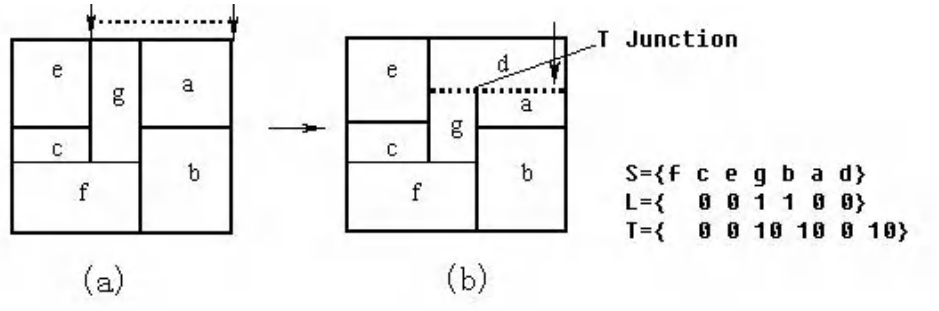


Fig. 2. From CBL to packing, insert corner block d to the corner and it covers one T junction so that it covers g and a

3 Room Resizing

Our algorithm is composed of four steps: (1)Determine maximum move space of each block, (2)try moving some block within its maximum move space, (3)applying traditional buffer insertion algorithm and (4) repeating 1,2,3 to search for an optimized solution. We will explain in details in the following subsections.

3.1 Determine Dead Spaces and Associate Them with Blocks

Dead space is defined to be space not held by blocks. During process of CBL packing, after placing the corner block from the top direction(L=0), all its covered blocks' top dead-spaces can be determined. Similarly, placing blocks from the right direction(L=1) will determine its covered blocks' right dead space. The

blocks on right boundary and top boundary are not covered by blocks and their dead spaces are settled with help of top boundary and right boundary. An example is shown in Fig. 3. In Fig. 3(a), corner block covers A so that X is A's top dead space. In Fig. 3(b), corner block covers D, C and A so that Y is A's right dead space and Z is C's Right dead space.

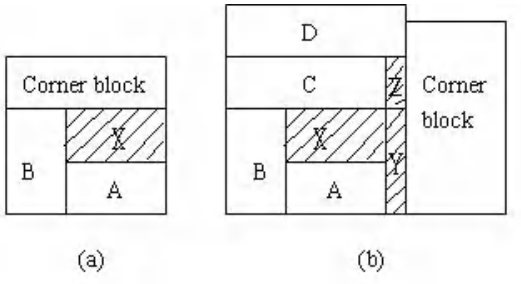


Fig. 3. While placing a corner block, its covered blocks' dead spaces could be determined

After the process of CBL packing, we not only determine all the dead-spaces, but also cut dead-spaces into rectangles naturally. At the same time, each dead-space is associated with a block and none of them intersect with each other. A block together with its dead spaces is defined to be this block's room. Since CBL is left-bottom packed, dead-spaces of a block are either on the top or on the right or both initially. Only moving blocks or redistributing dead-spaces can make a dead-space be on the left or bottom of its associated block. A more detailed description can be found in [4].

3.2 Vertical and Horizontal Constraint Graph

Building constraint graph during CBL Packing is a necessary step for our algorithm because constraint graph is a most efficient way to determine blocks' relative position. Firstly, we build a start vertex S and an ending vertex T. During CBL packing process, if the corner block covers from the top direction, we draw an edge from covered blocks to the corner block in vertical constraint graph, at the same time, we draw edges to the corner block from the blocks which lie left to the left-most covered block. If there are no blocks that lie left to left-most covered block which means the left-most covered block is on the left boundary, we draw an edge from S to the corner block. New blocks coming from right direction could be done similarly. In the end, we draw an edge from each block on top/right border to T. Considering edge-cost, we define it to be the dead space height/width between two neighboring blocks. An example is shown in Fig. 4, A comes from top and covers B and C, so we draw C->A and B->A in vertical graph in Fig. 4(a). C is the left-most covered-by-A block. E and F are Left to C, so we draw E->A and F->A in horizontal graph in Fig. 4(b). The only non-zero-cost edges are C->B, B->T.

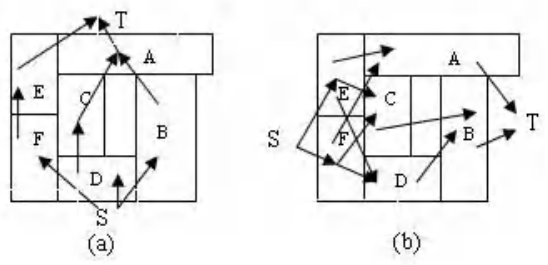


Fig. 4. Vertical(a) and horizontal(b) graph for a packing generated through a CBL

3.3 Maximum-Allowable Move Space (MAMS)

Since each block can be moved in four direction, up, down, left, right, so we keep a note of how much distance a block can be moved in one direction while the total area and topology remain unchanged.

Definition 1. *A block's maximum allowable move space on the right direction (MAMSR) is the maximum distance a block can be moved right, during which the other blocks that block the way may have to be pushed, but the total area and topology remain unchanged.*

The MAMSU, MAMSD, MAMSL, MAMSR which corresponds to the maximum-allowable move space on up/down/left/right direction can be defined similarly. An example is shown in Fig. 5. MAMSR of B is 4 because B can be moved right for distance 4 which pushes C, D and E, the total area and topology is unchanged. Placement after this move is shown in Fig. 5(b). After we move B, MAMSR of B is decreased to 0. Take D for another example, in Fig. 5(a), MAMSR of D is 7, after pushed by B, in Fig. 5(b), MAMSR of D is decreased to 3, but MAMSL of D is increased to 4 from 0.

Initially after CBL packing, each block's MAMSL and MAMSD is zero since we are packing them to left-bottom and none of them can be moved left or down.

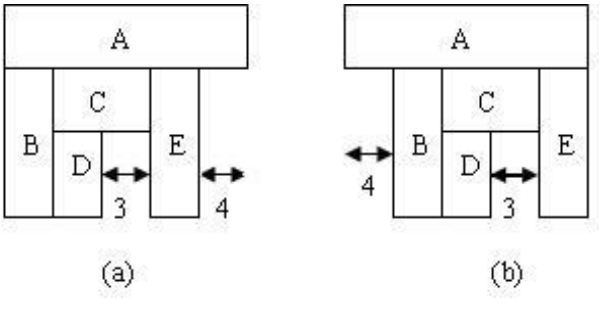


Fig. 5. An example for explaining MAMSR and MAMSL

MAMSR and MAMSU have to be calculated through constraint graph, a formula is shown below to determine MAMSR of block A.

$$MAMSR[A] = Min\{MAMSR[B] + edge_cost(A, B)\} \quad \text{with} \quad (A, B) \in E_h \tag{1}$$

$(A, B) \in E_h$ means that edge A->B is on the horizontal graph. This is a recursive algorithm to determine each block's MAMSR starting from blocks on the right border to left. Take Fig. 5(a) for example: MAMSR[E]= 4, MAMSR[A]=0,

MAMSR[C]=MAMSR[E]+ edge_cost(C,E)=4,
MAMSR[D]=MAMSR[E]+edge_cost(D,E)=7,
MAMSR[B]=MIN(MAMSR[C],MAMSR[D])= 4.

3.4 Room Resizing

After collecting information from the above algorithms, we have got enough knowledge for our next step, resizing room. Of course, we move blocks to achieve these. Before a block move, we must determine if this move is valid.

Definition 2. *A block move is a valid move if that, after moving this block which may push other blocks, the total area and topology could remain unchanged.*

Theorem 1. *A move is a valid move if and only if the distance is less than MAMS of the corresponding direction of that block.*

Proof. Take a right directed move for example without loss of generality, firstly, MAMSR is the cumulative distance a block can be moved right due to the dead spaces that lie on the right. If the distance exceeds this value, the blocks rightmost may be pushed out of the right boundary which causes the area to be enlarged; on the opposite, if there are enough cumulated dead spaces on the right to handle this move, the total area could remain unchanged. After we move the block right, MAMSR is decreased by that distance and MAMSL is increased by that distance. For example in Fig. 5(a), move B right 3 and Move D right 6 are both valid move, however Move B right 8 or Move C right 5 are both invalid moves. Moving any block left will be invalid because all blocks' MAMSL is zero. Since dead spaces are the only spaces for buffer sites, the ultimate objective of our block moving and room resizing is to redistribute dead spaces to make it more suitable for buffer requirements. Room resizing and dead space updating algorithm is shown as follows which takes moving right as an example.

```
1.  MoveRight (block A, distance i)
2.    If(A's right dead space is less than i)
3.      Foreach block B that lies right to A
4.        If( i >edge_cost(A,B))
5.           MoveRight(B, i-edge_cost(A,B));
6.        End if
7.        Allocate B's left dead-space
             to its left blocks(including A);
```

```
8.      End for.
9.    End if.
10.   Decrease width of A's right dead space by i;
11.   Increase width of A's left dead space by i;
12.   Update coordinates of block A;
13.   Update edge cost in horizontal graph;
```

This algorithm is also recursive that if a block wishes to move right, its right dead space width must be greater than the distance it wishes to move. If this is the fact, then moving will be easy and algorithm starts from step 10; if not, we have to push blocks that lie right and this will give this block larger room and enough right dead-space. We note that in step 7, giving left dead space to its left blocks will cause all its left blocks' right space to be broadened including "A" so that "A" will have enough right dead space for move and step 10 is able to go on.

3.5 Searching for Optimal Solution Using Simulated Annealing

Our final searching algorithm is based on Simulated Annealing (SA) to make each move effective for satisfying timing constraint. We start with an initial packing which is generated through simulated annealing(this is the first phase traditional simulated annealing which specialized on minimizing area and total wire length and it is quite different from the simulated annealing step in our algorithm). Based on this packing, we start to try moving blocks to optimize timing performance. We randomly find a block not on critical path, randomly select a direction it can be moved, and move it at a random distance which does not exceed corresponding MAMS. Then we update dead spaces and update the nets affected by the blocks having been moved. To analyze whether this move is an optimized move, we apply traditional buffer insertion algorithm on this packing. [3] displays details for the buffer-insertion algorithm which firstly calculate the number of buffers each net requires, get feasible region for each buffer and intersect buffer's feasible region with dead spaces. If the intersection is empty, the buffer will fail to be inserted. Only nets with no buffer need or with all buffers successfully placed is taken as timing-constraint-satisfied. So the effectiveness of each move is evaluated by whether it could make more nets meet timing constraint. With the random move of each blocks, the final result through simulated annealing process could minimize the number of nets which violate timing closure. The SA algorithm is shown below.

```
1.   Find an initial packing which satisfies basic requirement
        such as area and wire length;
2.   Draw constraint graph and determine each blocks MAMS
        and dead-spaces;
3.   While(Temperature>min_temperature)
4.      Randomly select a block not on critical path and a
          direction whose corresponding MAMS is greater than zero;
```

```
5.      Move that block on that direction at a random distance
          less than MAMS;
6.      Update MAMS, dead-spaces and nets information;
7.      Apply buffer allocation algorithm;
8.      If optimized accept it; else accept with a probability;
9.      Decrease temperature;
10.   End while
```

4 Experimental Result

The buffer planning and optimization algorithm have been implemented on SUN Ultra-SPARC and benchmarks are MCNC and CSTC(generated by duplicating ami33 and ami49 blocks and nets) benchmarks. The values for parameters are based on a 0.18m technology in the NTRS97 roadmap[11]. Since we are concentrating on two-pin nets, we decompose each net to two-pin nets using minimum-spanning tree. Because these benchmarks have no timing information and are not fit for direct buffer evaluation, we compute optimal delay Topt under optimal buffer insertion [10] (the minimized delay if we insert enough buffers on optimized locations) and assign a constraint delay 1.05 and 1.20 times Topt to the net. We choose this delay assign method so that each net's timing is possible to be satisfied and some buffers are often necessary to achieve this . Since the timing closures are randomly generated, it is not easy to compare it directly with result shown in [2]. However, we have implemented all these algorithms to make comparison in Table 1. The first column "No blocks moved" displays results after applying buffer allocation algorithm on the initial packing after CBL without optimization. The second column "DS redis. within room" displays results on packing generated by algorithm introduced in [2] which moves block within its room. The third column displays results on packing generated by our algorithm which redistributes dead-space based on room resizing. "Met/Total" is the number of nets satisfying timing constraint compared to the total number of nets, "Time" is the time consumed by each algorithm and "Improved" is the improving percent of our algorithm compared to the other algorithms. From the table below, we could see that our algorithm has an average improvement of 3.3% compared to algorithm without optimization and has an improvement of 1.58% compared to algorithm introduced in [2].

Table 1. Experimental Results

	No blocks moved		DS redis. within room		Our algorithm		Improved
	Met/Total	Time	Met/Total	Time	Met/Total	Time	Perc%
AMI33	340/363	<1s	348/363	6s	353/363	19s	3.8%/1.4%
AMI49	443/545	<1s	454/545	15s	463/545	72s	4.5%/2%
CSTC132	1313/1434	1s	1329/1434	52s	1351/1434	120s	2.9%/1.6%
CSTC199	1898/2111	1s	1913/2111	64s	1938/2111	160s	2%/1.3%
Average							3.3%/1.58%

5 Conclusion

In this paper, on the base of dead space redistribution buffer-insertion algorithm introduced in [2], we extend it by resizing rooms to improve number of nets that satisfy timing closure without violating topology and total area. Our algorithm provides steps for determining a block's maximum allowable move space and organize the dead spaces systematically so that buffer planning could be improved gradually with simulated annealing. Experimental results show that our algorithm is efficient for resolving the problem of buffer planning. Additionally, our algorithm is fit for IP based SOC planning since it often requires improving timing performance without violating topology. However, because a move may cause many blocks to move and room resizing and dead space update is more complicate, a faster moving solution and faster buffer evaluation method is in great need, which will be our future work.

References

1. X.L. Hong, S. Dong, G. Huang, et al: Corner block list representation and its application to floorplan optimization. IEEE Trans. on Circuits and Systems II: Express Briefs, Vol. 51, No. 5 , 2004, pp.228 - 233.
2. S. Chen, X.L. Hong, S.Q. Dong, Y.C. Ma: A buffer planning algorithm based on dead space redistributio. ASP-DAC, 2003.
3. J. Cong, T. Kong, and D. Z. Pan: Buffer block planning for interconnect-driven floorplanning. IEEE/ACM ICCAD, 1999.
4. Y.C. Ma, X.L.Hong, S.Q. Dong, S. Chen, Y.C. Cai: An Integrated Floorplanning with an Efficient Buffer Planning Algorithm. ISPD, 2003.
5. P. Sarkar, C. K. Koh: Routability-driven repeater block planning for interconnect-centric floorplanning. ISPD, 2000.
6. X. Tang and D.F. Wong: Planning buffer locations by network flows. Intl. Symp. Physical Design, 2000, pp. 180-185.
7. F. F. Dragan, A. B. Kahng, et al: Provably good global buffering by multiterminal multicommodity flow approximation. ASP-DAC, 2001.
8. C. W. Sham, F. Y. Young: Routability driven floorplanner with buffer block planning. ISPD, 2002.
9. F. Ragiq, M. C. Jeske, H. H. Yang, N. Sherwani: Integrated floorplanning with buffer/channel insertion for bus-based microprocessor designs. ISPD, 2002.
10. C. J. Alpert and A. Devgan: Wire segmenting for improved buffer insertion. Proc. Design Automation Conf, pp. 588C593, June 1997.
11. Semiconductor Industry Association, National Technology Roadmap for Semiconductors, 1997.

Analyzing the Performance of Mesh and Fat-Tree Topologies for Network on Chip Design

Vu-Duc Ngo, Huy-Nam Nguyen, and Hae-Wook Choi

System VLSI Lab Laboratory, SITI Research Center, School of Engineering
Information and Communications University (ICU),
Yusong P.O. Box 77, Taejon 305-714, Korea
{duc75, huynam, hwchoi}@icu.ac.kr

Abstract. The demand of integration of many heterogeneous semiconductor intellectual property (IP) blocks has been introducing a new chip design paradigm so called on chip network. This paradigm promisingly offers a packet switched network among IPs to reduce the main problems in the very deep sub micron technologies that arise from non-scalable global wire delay, failure to achieve global synchronization, errors due to the signal integrity, non-scalable bus based functional interconnection, etc. This paper introduces interconnected or switched network topologies and also analyze their performances in terms of communication protocol related to the issues such as routing strategy, buffer size, routing algorithm , etc. The above mentioned evaluations are done by utilizing the tool that has been widely used in the research domain of computer network design, so called NS-2.

1 Introduction

The continuation of scaling down of transistor size is leading to the fact that a large number of IPs can be integrated on a single chip. This integration, according to [1], gives rise to new challenges. The most prominent issue allied with the mass integration design comes up with non-scalable global wire delay [2]. Though sub-micron technology creates the good chance to reduce the gate delays, global wire delays significantly increase or remain constant by including repeaters. This matter also leads to the high consumption of silicon area and power. The other critical concerns related to forthcoming technology and current NOC design methodology are global synchronization and non-scalable bus based functional interconnection (especially when the number of integrated IPs is numerous). To overcome the above mentioned problems, we can propose the use of switching based topologies that have been extensively used for computer network to integrate IPs. Another word, the on-chip topology resembles the interconnect architecture of high performance parallel computing systems [3]. By using this design approach, the communications between IPs can be the packet based. We also realize that the needs of global wire can be omitted. The wires are now used only as the connections between switches. Also, we do not any longer worry about global synchronization due to decoupling of the processing

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 300–310, 2005.
© IFIP International Federation for Information Processing 2005

nodes. To get over the induced noise caused by unreliable physical channels, the packet based error control mechanisms can be applied. The two promising on chip network topologies have been mostly proposing are Mesh and Fat-Tree, respectively. The comparison of performance evaluation of the two topologies with respected to physical constraints was made by Petrini and et al. [4]. The comparison showed clearly that the Fat-Tree topology proposed firstly by Leiserson [5], with several merits such as:

– *Scalability; Reliability; Regular and simple topology; High throughput; Good performance,*

has superior performance to that of the Mesh topology. To verify these above mentioned merits in terms of on chip network topology design approach, we need to carry out the high level simulation Fat-Tree topology in the scenario of on chip network implementation. However, the high level simulation of on chip network is still in its infancy. Choosing simulation tool and defining the physical constraints for it are now at the beginning stage. There is no gate level simulation tool, which is known as the best method to evaluate the physical behavior of SOC design, can be developed to adapt to on chip network design and verification. To conquer this limitation we choose network simulation tool to simulate the on chip network with respect to high level approach. Among network simulation tools, NS2 [6] emerges to be the best solution to simulate the high level of on chip network architecture. The NS2 is widely used in the research for design and evaluation of computer network domain. We take the advantage of NS2 in terms of communication protocol (OSI protocol) to simulate and verify the performance of our proposed interconnection architecture for on chip network's application, including topologies selection, the design of router, and routing schemes, as well as queuing schemes. In [7] and [8], the authors proposed the Mesh and Fat-Tree architectures for on chip network topology design, respectively. However, the performances of these architectures are not yet mentioned in terms of high level of topology evaluation. In this paper, we report several experiment results of general NoC and H.264 video decoder design based on the simulation of on chip network using NS2 for Mesh and Fat-Tree architectures with the relevant routing and queuing schemes. By using these simulation results with certain comparisons, we propose to use Fat-Tree as on chip network interconnection topology. Our paper is organized as follows: The interconnection architectures and routing schemes are introduced and compared in section II. The NS2 and its mapping to on chip network is conducted in section III. The simulation results and their comparisons are analyzed in section IV. Finally the conclusion and future work will be given in section V.

2 Interconnection Architectures and Routing Schemes

2.1 Interconnection Architecture

Mesh architecture: The $k-ary\ d-dimensional$ Mesh architecture is built by its dimension d and radix k. This follows that the total number of switches

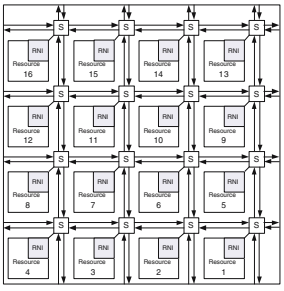


Fig. 1. 4-ary 2-dimensional Mesh topology

is k^d. The k^d switches are organized in an $d-dimensional$ grid such that there are k switches located in each dimension and wrap-around connections. Since the number of IPs that can be connected to one switch is $d-1$, it follows that the total number of mounted IPs is clearly calculated by

$$N_{Mesh} = k^d(d-1). \tag{1}$$

We denote b as the unidirectional bandwidth, the total bandwidth of the network is obtain by

$$B_{Mesh} = 2k^d b. \tag{2}$$

Moreover, this architecture offers a simple connection scheme. Therefore, the Shortest Path routing is mostly applied on it. Also the number of the mounted IPs is relatively high compared to the total bandwidth. So that the performance of Mesh architecture is not high.

The Fig.1 shows an example of $4-ary\ 2-dimensional$ Mesh architecture. We can see in this figure the number of switches and the number of the mounted IPs are both equal 16.

Fat-Tree architecture: The Fat-Tree is an indirect interconnection network based on a complete binary tree, which was proposed by Leiserson [5]. There is a significant difference from the traditional tree in computer science, Fat-Tree copies image from real tree that it gets as thicker as it approaches the root. A set of processors are located at the leaves of the Fat-Tree and each edge of

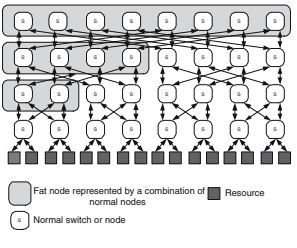


Fig. 2. 4-ary 2-dimensional Fat-Tree topology

the underlying tree corresponds to a bi-directional channel between parent and child. The bandwidth of the Fat-Tree increases as it goes closer to the root. This feature makes Fat-Tree to be sufficiently applicable with advanced routing algorithm as the Distance Vector instead of Shortest Path. Furthermore, since the depth of the tree is calculated by the dimensional metric plus 1, it follows that the size or the number of IPs that can be mounted on one certain Fat-Tree topology depends on the depth of the tree. As the depth of the tree increases, the number of IPs as well as the total capacity of the network gets bigger. These two significant characteristics lead Fat-Tree having better performance in comparison with Mesh architecture. More analytically, we introduce some calculations in $k-ary\ d-dimensional$ Fat-Tree architecture, which is shown in Fig.2 (case of $k = 4$, $d = 2$).

As can be seen in the Fig.2 the fat nodes are the switches that are located in the higher degrees of the tree and they can be considered as the combination of $2^{(t_d-1)}$ single nodes, where t_d presents the node's degree on the tree. The number of mounted IPs $N_{Fat-Tree}$ and the number of switches $S_{Fat-Tree}$ are straightforwardly calculated as follows:

$$N_{Fat-Tree} = k^d, \tag{3}$$

$$S_{Fat-Tree} = k^{d-1}d, \tag{4}$$

respectively. Therefore with the unidirectional bandwidth defined by b, the total bandwidth is presented by

$$B_{Fat-Tree} = 2k^d db. \tag{5}$$

We carry out the comparison of this above total bandwidth with the case of the $k-ary\ d-dimensional$ Mesh's total bandwidth, which is shown in (1). With $d \geq 2$, the total bandwidth of the $k-ary\ d-dimensional$ Fat-Tree architecture is equal d times of the total bandwidth of $k-ary\ d-dimensional$ Mesh architecture. Since more available total bandwidth is offered, it follows that the more propagation delay can be decreased. This fact is very good factor to consider Fat-Tree as a appropriate candidate for on chip network topology. However, the designers have to accept the higher network cost or complexity.

2.2 Routing Schemes

Shortest Path: The distributed Shortest Path routing scheme [12] is an iterative procedure to find out the routing table which contains the shortest paths from all the nodes of the network to each common destination which also belongs to the network. Let one of the common nodes be defined as x. In order to keep track of the shortest paths, we label each non-destination node i with a pair (η, d_i^x) where d_i^x represents the current iteration for the shortest distance from the node to x and η is the number of next nodes along the currently computed shortest path. The iterative algorithm is presented as the following 4 steps:

– Initialization: Set the initial value for the destination node x:

$$d_x^x = 0, \tag{6}$$

and label all other non-destination nodes $(\bullet, +\infty)$.
– Updating: Update d_i^x for each non-destination node i by checking the current value of d_j^x for each neighbor node j and carrying out the operation:

$$d_i^x \leftarrow \min_j [d_j^x + l(i,j)], \tag{7}$$

where $l(i,j)$ denotes the length of the link from node i to node j.
– Node j's label is completely updated by replacing the metric η by the number of the neighbor nodes which minimize the (7).
– Step 2 is repeated at each node until no further changes occur.

This above algorithm is terminated with the output of routing table. By this table the data packet with a destination node identifier in its header can reach it's desired destination node with a certain route that contains specifically preferred neighbors assigned by predetermined routing table. This fact significantly affects network performance due to its static characteristic. We will discuss this issue in the IV section. However, this routing algorithm is still widely applied over others for implementation within operational networks due to its simplicity.

Distance Vector: In this context we reintroduce the Distance Vector routing algorithm [12] that was proposed relying on Distributed Bellman-Ford algorithm. In this algorithm, each node i maintains a set of distances d_{ij}^x where j varies over the neighbors of i. Moreover, node i treats its neighbor k as the next intermediate node that the data packet visits before reaching the given destination x if:

$$d_{ik}^x = \min_j \{d_{ij}^x\}. \tag{8}$$

The succession of next intermediate nodes selected in this above method leads to x along the shortest path. To keep track of the distance estimation up-to-date, each node monitors the length metrics of its available outgoing links. It periodically broadcasts its current estimate of the shortest distance to every other node inside the network to each of its neighbors . By checking out the available outgoing links, this algorithm outperforms to that of Shortest Path algorithm. The comparisons will be performed by simulation in section IV.

3 NS2 and Its on Chip Network Mapping

The network simulator, NS2, with its capabilities of describing network topologies, network protocols, routing algorithms, packet scheduling schemes, as well as traffic generation methods, has been broadly used in the field of computer network design and simulation. Referring to the article [9], the authors used NS2 to simulate the Mesh architecture with dimension of 5x5 for on chip network topology. However, only the simplest cases of routing algorithm, routing strategy,

queuing algorithm, transmission protocol are defined as simulation constraints. In this paper, we not only carry out the simulations for $2-dimensional$ Mesh architecture with the advanced issues of those mentioned constraints but also for $2-dimensional$ Fat-Tree architecture to make a comparison and choose a good candidate for on chip network topology.

For both of Mesh and Fat-Tree topologies, one IP, called resource, is always connected to one switch, called router. The interface between IP and switch is called Resource Network Interface (RNI). The functions of RNI cover 4 upper layers of the internet 5 layers model. The physical layer is functioned by the linkages between resources and routers (Rs2Rt) or between routers and routers (Rt2Rt). In NS2 environment the transmission protocol can be defined as UDP or TCP. The exponential traffic generator model is applied. The routing strategies can be: Shortest path, Dynamic or Distance Vector (DV), according to the choice of two architectures (Mesh or Fat-Tree). Since IPs (resources) of on chip network platform can be CPU, RAM, DSP, FPGA, USB, etc, the demand of communication loads originated from each IP are different. This leads us use the fair queuing scheme like Stochastic Fairness Queuing (SFQ) instead of DropTail introduced in [9]. The uses of FQ and SFQ for our mentioned architectures achieve overperformance to that of the DropTail in terms of drop probability, delay, as well as throughput. The Table. 1 shows the summary of mapped NS2 constraints for our experiments on the two mentioned architectures.

4 Simulation Results and Discussion

In this paper, we firstly simulate several cases with different constraints of routing and queuing algorithms as well as buffer sizes for the 4x4 Mesh and $4-ary$ $2-dimensional$ Fat-Tree architectures. In detail, we simulate these two architectures with Shortest Path and Distance Vector routing algorithms, also we attach the congestion control or Queuing algorithms such as DropTail, RED (Random Early Detection), DRR (Deficit Round Robin), and SFQ (Stochastic Fairness Queuing)to the switches of both architectures. Finally, we compare the simulation results of all simulated cases to choose the best combination of embedded routing and queuing algorithms for each topology. We also compare the

Table 1. Application of On chip network constraints to NS2

On chip network model	Applied NS2 constraints
Connections	Rs2Rt and Rt2Rs
Transmission protocols	UDP and TCP
Packet size	8 bytes
Buffer size	8, 16, 32 (packets)
Queuing schemes	DropTail, DRR, RED, SFQ
Routing schemes	Shortest Path, Distance Vector
Traffic generator	Exponential
Bisection bandwidth (Max)	8 Gbits (expected)

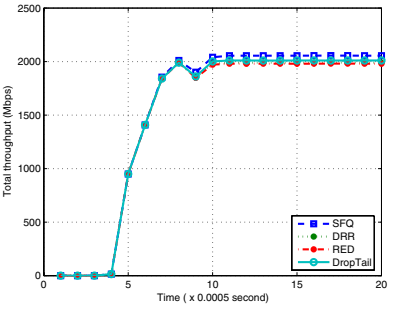


Fig. 3. Throughput of Mesh (Shortest Path) vs. Queuing Schemes

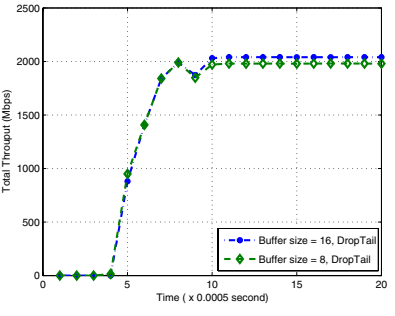


Fig. 4. Throughput of Mesh (Shortest Path) vs. Buffer sizes

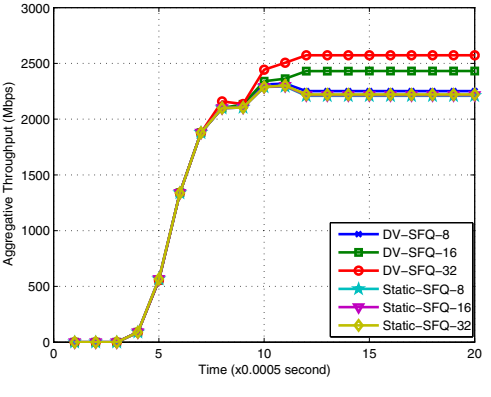


Fig. 5. Throughput of Fat-Tree (SFQ queuing schemes) vs. Buffer sizes and routing schemes

performances offer by two architectures in the cases of using the similar options of routing, queuing, as well as the buffer sizes. We already defined the on chip network model and it mapped constraints onto NS2 simulator in the Table. 1. Due to the limitation of the computer capability, we just limited the data rate

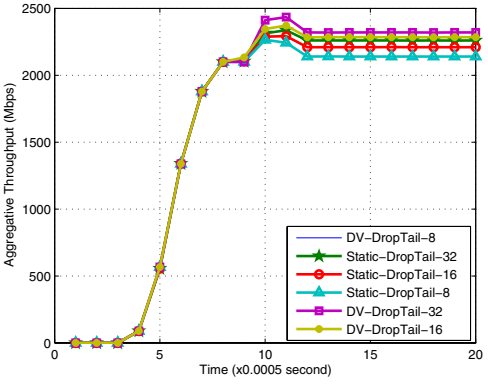


Fig. 6. Throughput of Fat-Tree (DropTial queuing schemes) vs. Buffer sizes and routing schemes

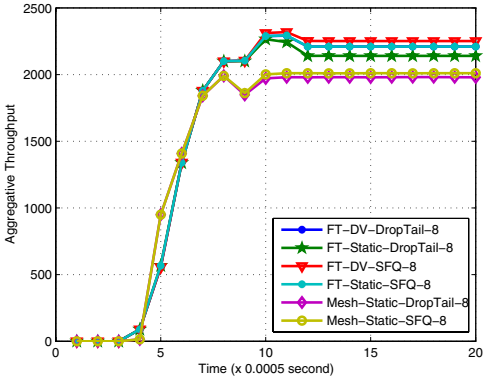


Fig. 7. Throughput Comparison of Fat-Tree (FT) and Mesh topologies

of each resource up to 200Mbps. The resources are activated to generate data packets randomly. The Fig. 3 and Fig. 4 depict the aggregative throughput of 4x4 Mesh topology vs. different options of applied queuing schemes and buffer sizes, respectively. As can be seen in the Fig. 3, with the same buffer size as well as routing scheme (Shortest Path), the use of SFQ queuing scheme offers us the best performance with respect to total throughput on 4x4 Mesh topology. The Fig.4 shows that the bigger buffer size is applied, the higher throughput is achieved. The throughputs of Fat-Tree topology with queuing schemes of SFQ and DropTail are depicted in Fig. 5 and 6, respectively. We realize that by applying Distance Vector routing algorithm to Fat-Tree topology, with its own merits as pointed out in previous sections, the system throughput can be increasingly achieved. In the last figure, Fig. 7, we compare Fat-Tree and Mesh topologies with the same buffer size of 8 packets in terms of system throughput related to different choices of routing and queuing schemes. We can see that the Fat-Tree topology with the embedded routing algorithm of Distance Vector and queu-

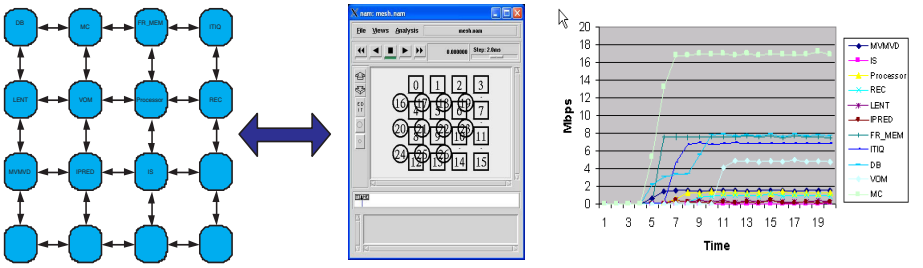
From/To	DB	MC	FR_MEM	IITIQ	LENT	VOM	Processor	REC	MVMVD	IPRED	IS
DB			7114								
MC	4475		345								
FR_MEM	2650	10652			63	4802		40			
IITIQ		5949						642			
LENT				6592			408		496		
VOM											
Processor	326	322		186	232	12		7	922	59	12
REC	321	13	27							145	
MVMVD							828				
IPRED								321			
IS			63								

Fig. 8. IPs and their connections (in kbps) of H.264 video decoder

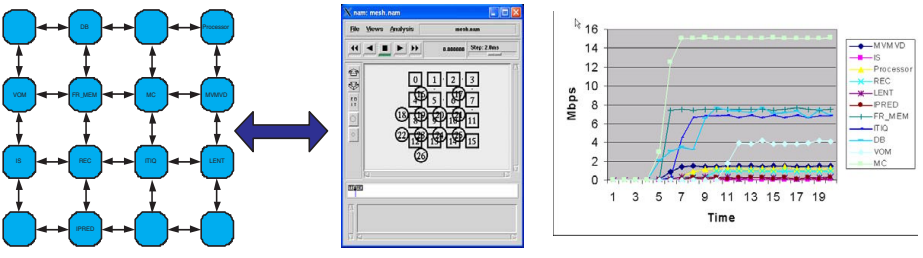
ing algorithm of SFQ gains highest throughput. In the stable period, this design achieves approximately 250Mbps higher to that of Mesh topology using Shortest Path and DropTail schemes [9].

More specifically, we implement the H.264 video decoder on those mentioned NoC architectures. The IPs and their connections represented by the directional data rates are defined by the table in Fig. 8. There are 11 IPs that are allocated onto 11 corresponding switches. To maximize the throughput transactions among the IPs, we adopt the Branch and Bound technique presented in [13] for the cost function of data rates. The results of optimum mapping shows that the IPs which are connected by the connections of high data rates should be allocated next to each others. This mapping reduces the drop packets that is very high rate if we allocate the mentioned IPs far away since huge number of packets from the original IP need to go through a lot of intermediate switches before reach the destination. In order to make this argument clearly, two mapping schemes of IPs onto the 4x4 Mesh architecture are simulated as depicted in Fig. 9. As seen in this figure, the throughputs of IPs of random mapping is shown by Fig. 9a are almost similar to that of the throughputs of IPs of the optimum mapping shown by Fig. 9b, but the corresponding bisectional bandwidth required for random case of 30Mbps is much more higher to that of 15Mbps for the optimum case. These two simulations are done with SFQ queuing scheme and DV routing scheme. The last experiment is carried out to show the superior performance of using Fat-Tree compared to Mesh architecture. As seen in Fig. 10, with bisectional bandwidth of 20Mbps, not only the system requirement of throughput is met but also the throughputs of IPs are almost similar to the case of random mapping of Mesh architecture with the bisectional bandwidth of 30Mbps.

In conclusion, we propose to use Fat-Tree topology with embedded Distance Vector routing and SFQ queuing schemes as on chip network switching core topology.

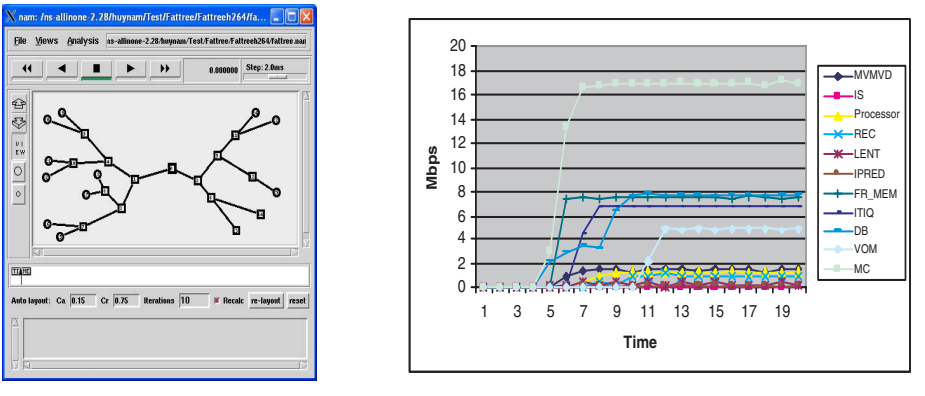


(a). Bisectional bandwidth of 30Mbps, SFQ, DV



(b). Bisectional bandwidth of 15Mbps, SFQ, DV

Fig. 9. IPs allocation schemes their throughputs (in Mbps) of H.264 video decoder with Mesh architecture: (a) random allocation, (b) optimum allocation



Bisectional bandwidth of 20Mbps, SFQ, DV

Fig. 10. IPs allocation scheme their throughputs (in Mbps) of H.264 video decoder with Fat-Tree architecture

5 Conclusion

In this paper, we compare the performance of Fat-Tree and Mesh architectures in the sense of on chip network design methodology. We also carry out the high level simulation of on chip network based on Mesh and Fat-Tree architectures using NS2 to verify our analysis. The simulations of the Mesh and Fat-Tree architectures are done with advanced routing algorithms and router queueing mechanisms for general NoC design as well as H.264 video decoder. Finally, by numerical and simulation results we can conclude that the Fat-Tree architecture is suitable for on chip network switching core. By utilizing Fat-Tree architecture and Distance Vector routing algorithm as well as Stochastic Fairness Queue algorithm, we can achieve the best combination for our design in terms of choosing on chip network topology. It has superior performance to that of Mesh architecture, in which Shortest Path routing algorithm is applied, with respected the system throughput.

References

1. L. Benini and G. DeMicheli,"Networks On Chips: A new SoC paradigm," IEEE computer, Jan, 2002.
2. M. A. Horowitz et al.,"The future of wires," Proceeding of IEEE, Vol. 89, Issue. 4 , pp. 490-504, Apr 2001
3. P. Guerrier, A. Grenier,"A generic architecture for on-chip packet-switched interconnectio,"Design automation and test in Europe conference, pp. 250-256, Aug 2000.
4. F. Petrini and M. Vanneschi,"Network performance under Physical Constrain," Proceedings of International Conference on Parallel Processing, pp.34 - 43, Aug 1997.
5. C. E. Leiserson,"Fat Trees: Universal networks for hardware efficient supercomputing," IEEE Transactions on Computer, C-34, pp. 892-90,1 Oct 1985.
6. Ns2: www.isi.edu/nsnam/ns/.
7. H. Kariniemi, J. Nurmi, "New adaptive routing algorithm for extended generalized fat trees on-chip," Proceedings of International Symposium on System-on-Chip, pp. 113 - 118, Nov 2003.
8. P. P. Pande, C. Grecu, A. Ivanov, R. Saleh, "Design of a switch for network on chip applications," Proceedings of the 2003 International Symposium on Circuits and Systems, pp. V-217 - V-220 vol. 5, May 2003.
9. Yi-Ran Sun, S. Kumar, A. Jantsch, "Simulation and Evaluation for a Network on Chip Architecture Using Ns-2," Proceeding of 20th IEEE Norchip Conference, Nov 2002.
10. D. Berozzi, L. Benini and G. DeMicheli,"Low-Power Error Resilient Encoding for On-chip Data Buses," DATE-International Conference on Design and Test Europe, Jun 2002.
11. J. Muttersbach et al., "Globally-asynchronous locally synchronous architectures to simplify the design of on-chip systems," Proc. 12th Annual IEEE International ASIC/SoC Conference, pp. 317-321, Sep 1999.
12. D. Bertsekas and R. Gallager, "Data Networks," Chapter 5., Second Edition, Prentice-Hall, Inc., 1992.
13. S.Murali and G.De Micheli, Bandwidth-Constrained Mapping of Cores pnto NoC Architectures, , DATE, International Conference on Design and Test Europe, 2004, pp. 896-901.

Hierarchical Graph: A New Cost Effective Architecture for Network on Chip

Alireza Vahdatpour[1], Ahmadreza Tavakoli[2], and Mohammad Hossein Falaki[1]

[1] Computer Engineering Department,
Sharif University of Technology
{vahdatpour, falaki}@ce.sharif.edu
[2] Electrical Engineering Department,
Sharif University of Technology
artavakoli@ee.sharif.edu

Abstract. We purposed a new Network on Chip (NoC) architecture called Hierarchical Graph. The most interesting feature of this novel architecture is its simple implementation process. Furthermore, the flexible structure of this topology makes it suitable for use in application specified chips. To benchmark the suggested architecture with existing ones, basic models of physical implementation have been extracted and simulated using NS-2. The results compared with the common used architecture Mesh show that HG has better performance, especially in local traffics and high loads.

1 Introduction

As predicted in [1], we will enter the *Ultra Large Scale Integration* (ULSI) design world with 45nm technology in 2010. Integration of multibillion transistors will be possible and device delays will get smaller. The dominant issue in design would be the interconnection between transistors.

The closer we get to large chip integration, the more we need smart design plans. Currently one of the most accepted plans is *System on a Chip* (SoC). In SoC methodology a number of IP-cores (FPGAs, mixed signal blocks, processors, memory blocks etc.) are placed together on a single chip, with proper systematic interconnections. Today the most common method of interconnection is BUS. While the number and complexity of cores are increasing in *Ultra Deep Submicron*, interconnections become more challenging.

Computer networks have been successful in handling the problems of managing large number of resources. This has led to a new paradigm in the design of SoC systems, called *Network on Chip* [2] which is compatible with *Globally Asynchronous Locally Synchronous* (GALS) design method. NoC design uses protocols similar to OSI reference model of networking [3], as the interconnection facility between IP cores. Currently, the most common network architectures for NoC systems are Mesh and Fat-Tree [4].

In this paper we have introduced a new architecture for NoC systems named *Hierarchical Graph* (HG). Some related work is addressed in the next section and

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 311–320, 2005.
© IFIP International Federation for Information Processing 2005

then various aspects of the proposed architecture are introduced in section 3. In section 4 some physical properties of the implementation of HG are discussed and then the performance of the new architecture is compared with Mesh through the simulation. Finally the results of simulations are presented in section 5.

2 Related Work

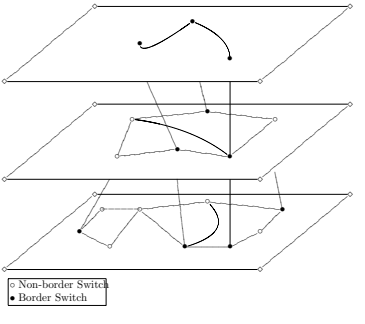
There are many research groups working on SoC systems from architectural point of view. In [4] Shashi Kumar et al. introduced a new methodology for designing mesh architecture for NoC. Lacking specific tools for evaluating networks on chip, the performance of Mesh and Fat-Tree were compared using Network Simulator (NS-2) in [5] and [6]. Some useful strategies were introduced by Daniel Wiklund et al. in [7] for benchmarking on chip networks.

3 Architecture

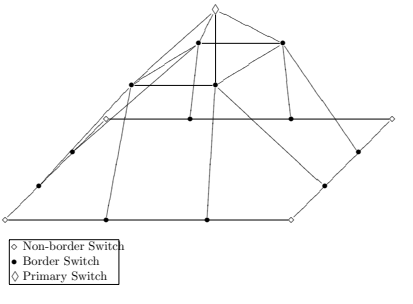
Hierarchical Graph (HG) network topology is designed specially for NoC systems to be compatible with Ultra Deep Submicron fabrication technologies.

Fat-Tree as a network topology has good performance, but its irregular and intensive interconnection lines make it impractical to be implemented on a chip. Although the throughput of HG is not as good as Fat-Tree, it can be implemented with a single metal layer because of its simple interconnection rules. Decreased ratio of switch per resource and simpler switch implementations decreases the implementation cost.

As the name suggests, HG consists of a hierarchical structure of plain graphs (see Fig.1(a)). In each level, there is a graph of interconnected switches, some of which named *Border Switches* are responsible for connecting the graph of this



(a) General Hierarchical structure of HG

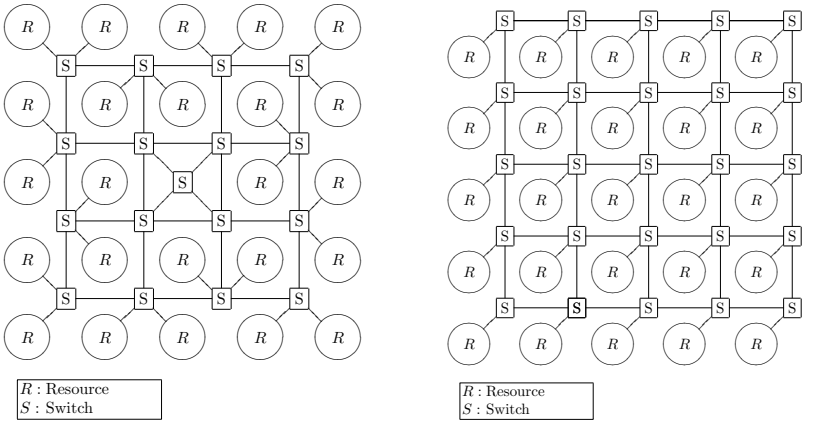


(b) Structure of proposed Square HG(SHG.)

Fig. 1.

level to the upper level graph. To facilitate physical implementation of interconnections, the graph in each level can have a regular shape e.g. square, which leads to *Square Hierarchical Graph* (SHG). The main focus in this paper is on SHG. The number and position of the border switches are flexible parameters of this architecture. A good choice of border switches in a SHG network would avoid diagonal links as shown in Fig.2(a). In the highest hierarchy, there is always a single powerful switch called *primary switch*. In HG the cores can be placed in different levels of the hierarchy according to their specific communication needs. For example the cores with higher traffic requirements such as central processing units or shared memories can be placed in higher levels to reduce latency and increase the overall performance of the chip.

To analyze performance parameters of HG we used a specific version of SHG. For N^2 general cores, such a SHG network uses $(N-1)^2$ switches with strict connection lines to make a symmetric graph with maximum performance and minimum latency. When N is odd, the central core is substituted with a high performance switch named primary to make the graph more symmetric. The network in Fig.1(b) which is a version of the SHG just mentioned consists of 17 switches, one powerful switch in the highest level, 4 ones in the second level and 12 switches in the third level. Four of the switches in the lowest level are not border switches and the rest are all border switches.



(a) Overview of SHG architecture (b) Overview of Mesh architecture

Fig. 2.

Simulations in section 4 demonstrate that, this type of SHG outperforms Mesh in many benchmarks while its number of switches is less than the equivalent Mesh (see Fig.2(b)). It is predicted that the complexity of switches in NoC systems would increase dramatically as the need for implementation of heavy weighted routing algorithms, large queues and queue management disciplines

increases. Thus the fewer number of switches would reduce the chip area and overall cost. According to [8], switches with less than five inputs and output virtual channels have simple implementation. So HG switches are not more complex in comparison to common switches used in Mesh.

4 Simulation

Simulations were used to compare the new architecture with existing ones from the performance point of view. Our intention was to prove the suggested architecture has no or little performance penalty while reduces costs. Among the two other architectures , Mesh and Fat-Tree, only Mesh is compared to HG. This is because of the implementation issues of Fat-Tree, discussed in 4.1. In the section 5 the results are compared and different aspects of performance issues of the new architecture are discussed.

Simulations were done using two different traffic models: random traffic and local intensive traffic. In each simulation scenario we used a Mesh of 25 cores which requires 25 switches and a SHG with 24 cores in which 17 switches are involved. The two measured performance indicators were "drop probability" and "overall packet delay". Drop probability is the ratio of the number of dropped packets to the total number of generated packets. For the overall packet delay, end to end delay of all delivered packets was measured and averaged.

The underlying transmission protocol used was UDP. UDP offers no congestion control and reliability. While these two issues are not covered well in NoC researches, TCP is not a good choice because of its high overhead, non predictable delay and other implementation issues. The use of an application specific congestion control can increase the overall performance. There should also be methods to detect and recover delivery failures either in application layer or network layer. These issues are still open to research. Since we are just interested in the architectural performance evaluation, we can safely use UDP for all architectures.

The two traffic models discussed before differ in the choice of sender-receiver pairs. In the random traffic model, each node sends data to another random node. Since each node has no more than one traffic sink and source, this is equivalent to a uniform distribution of sink and source pairs among the cores. While this traffic model can indicate some performance features of any architecture, in practice the locality of traffic is important. For the local traffic model we chose source and sink pairs in a way that 75% of traffic is local. The rest 25% was distributed randomly among the remaining nodes.

The traffic source behavior is also of special importance, while the particular traffic behavior can greatly depend on the specific application, exponential traffic seems most suitable for the simulation purpose. All simulations were also repeated with randomized Constant Bit Rate (CBR) to cover a wider range of traffic behaviors in real applications.

4.1 Modeling Simulation Parameters

For simulation of any network architecture for NoC, we need system level modeling of different chip implementation parameters such as switch delays, link maximum bandwidth and delay of links.

Our modeling parameters are based on the ITRS 2004 data for 45 nm technology predicted to come to market in 2010 [1]. For modeling purposes we will use the top level global interconnection parameters. As predicted in [1], these interconnections will have RC delay about $143ps/mm$ to $189ps/mm$ (the exact amount of delay depends on scattering effects). Also the intrinsic delay of each NMOS is considered to be about $0.39ps$ to $0.98ps$, (the exact amount of delay depends on performance and power efficiency of the device). With these delay values for NMOS the delay of a NAND gate would be about $9.88ps$ to $24.8ps$. The number of maximum parallel lines that can be used to increase bandwidth of a link is also important. This depends on global links wiring pitch size which is predicted to be about $205nm$.

In Fig.2(a) there are 24 cores, one primary switch and 16 ordinary switches. If we assume the cores and primary switch to be about $2.5 \times 2.5mm^2$ and the ordinary switches about $0.5 \times 0.5mm^2$, in a chip area of $1.5 \times 1.5cm^2$, the length of the links between ordinary switches would be $2.5mm$ and the length of the links between cores or primary switch and ordinary switches is $0.1mm$. The overall length of the links is smaller than $10cm$, so using the given pitch size there could be more than 5000 parallel lines in each link. For simplification of modeling only 1024 parallel lines have been used. Using $\sim 200ps/mm$ delay for links, delay of a link between two switches will be about $500ps$ and the links from a switch to a core would have about $20ps$ delay. The switch implementation is based on [8]. The delay of this switch could be modeled with 5 series of NAND gates estimated about $50ps$. The delay of each *Resource Network Interface* (RNI) is about half of the delay of an ordinary switch and thus about $25ps$.

For a signal transmitted from a RNI and received by a switch, the overall delay consists of RNI delay, link delay and half of a switch delay which is totally about $70ps$. This means $550ps$ switch to switch delay. For calculating switch to link bandwidth, $550ns$ delay is considered leading to $2GHz$ bandwidth for each line. So 1024 parallel lines bundled in a link would have $2048GHz$. Using this method the bandwidth of the link between a switch and a core is $51.2GHz$.

In Mesh topology (Fig.2(b)) core to switch links and switch to switch links are the same as in SHG. Also RNI and Switches have approximately the same structure as in SHG. So the same model as SHG is used for Mesh.

In Fat-Tree the connections cannot be implemented on a single metal layer and its long link distances in real chips make the bandwidth of links very poor in comparison to the two other topologies, Mesh and HG.

4.2 Simulation Experiment

All simulations were done using the publicly available Network Simulator (NS-2) [9]. Both a traffic sink and a traffic generator were attached to each node. In

other words each node could both send and receive data which is near to what is expected in real applications. Since the selected Mesh network had one core more then SHG, the central core of Mesh was not involved in simulations.

To reduce high variations in random traffic model we had to repeat each simulation and use the average result.

Since NS-2 is designed for computer networks, the bandwidth and delays proposed for a NoC system are not suitable for simulation. We scaled down all the estimated quantities by dividing all bandwidth values by a factor of 10240 and multiply delay values by a factor of 10240, so all simulation results are in terms of scaled quantities.

5 Results

As mentioned before the results of simulations are average packet delay and drop ratio for Mesh and SHG, using two traffic models and two traffic source behaviors. The results are presented here according to the traffic model. In the following graphs delay and drop probability are shown versus traffic load generated by a single traffic source.

5.1 Random Traffic Model

Although the random traffic model is not what is expected to happen in real applications it can provide a good measure for comparing the two architectures.

The use of random CBR traffic generators in the random traffic model for SHG and Mesh resulted in Fig.3 and Fig.4. The graph in Fig.3 shows average packet delivery delay versus traffic load of any traffic source and the graph in Fig.4 depicts packet drop ratio vs. traffic load. While in SHG, packet drop begins near $85Mb/s$ for a source, in Mesh packet drop begins from $90Mb/s$. No NoC

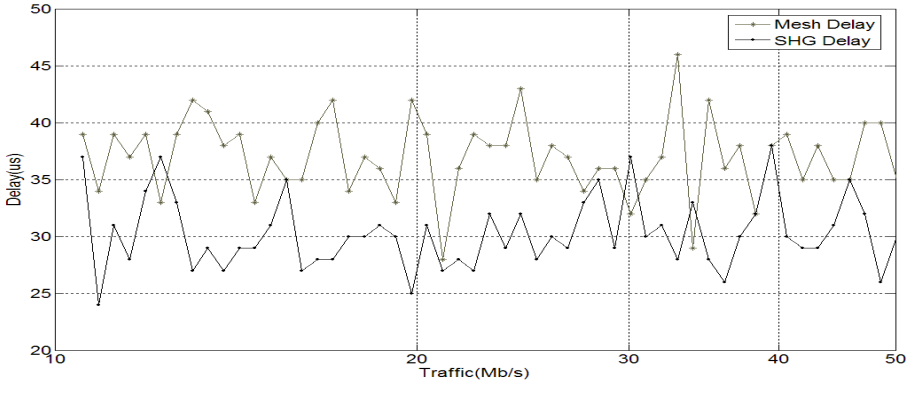


Fig. 3. Packet delay vs. traffic load of a random CBR source in a random traffic network (random traffic model)

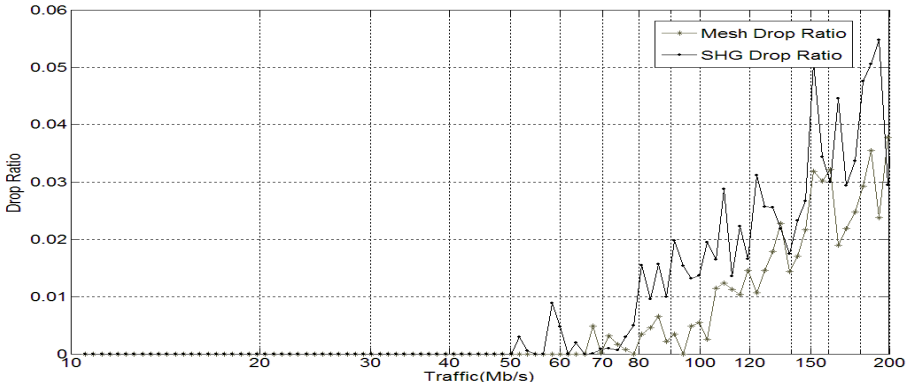


Fig. 4. Drop probability vs. traffic load of a random CBR source in a random traffic network (random traffic model)

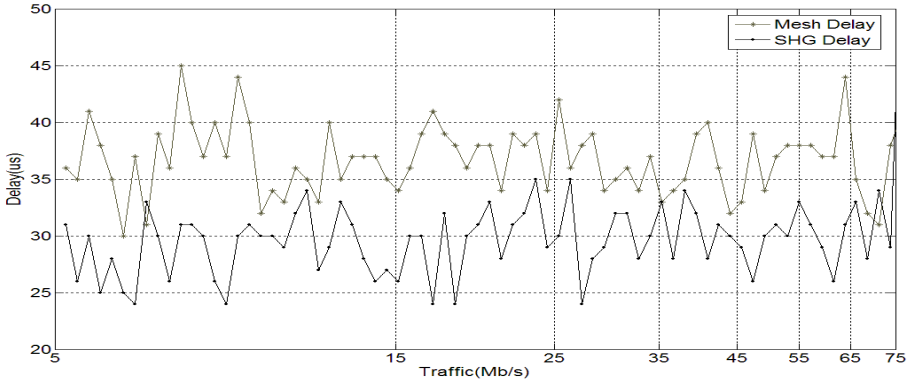


Fig. 5. Packet delay vs. traffic load of an exponential source in a random traffic network (random traffic model)

system can be used in traffic loads near these values but for comparison purposes we continued the simulations beyond these points.

Fig.3 shows that the overall packet delay in SHG is less than Mesh by a constant factor. On the other hand packet drop ratio is less in Mesh.

The exponential traffic source simulations lead to similar results in Fig.5 and Fig.6.

The exponential traffic source is expected to be more realistic as a traffic source. As the graphs show, the results for this type of traffic source have very little difference from the outcome of random CBR traffic source simulations.

The results of random traffic model indicate that SHG performs better when considering overall packet delay. On the other hand SHG is defeated by Mesh when considering drop. The point where considerable packet drop begins is the same in both topologies and after that the drop rate is higher for SHG. After the

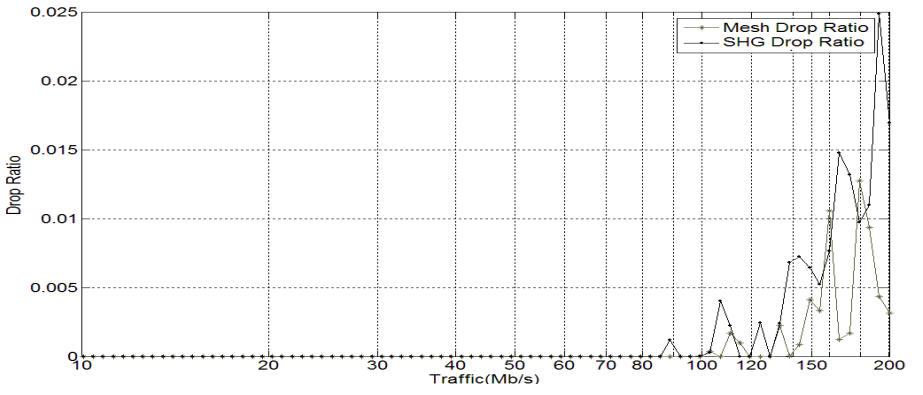


Fig. 6. Drop probability vs. traffic load of an exponential source in a random traffic network (random traffic model)

packet drop starts the network is considered unusable. Thus the most important section of all simulations is where the network is working without any drop.

5.2 Local Traffic Model

The local traffic model resembles real applications to a greater extent than random traffic. Thus the results of this section are of more interest and importance. Besides the more interesting nature of local traffic model, the absence of random variations makes the graphs more readable especially for the exponential simulation.

Fig.7 shows the results of simulating random CBR sources in SHG and Mesh in a local traffic model.

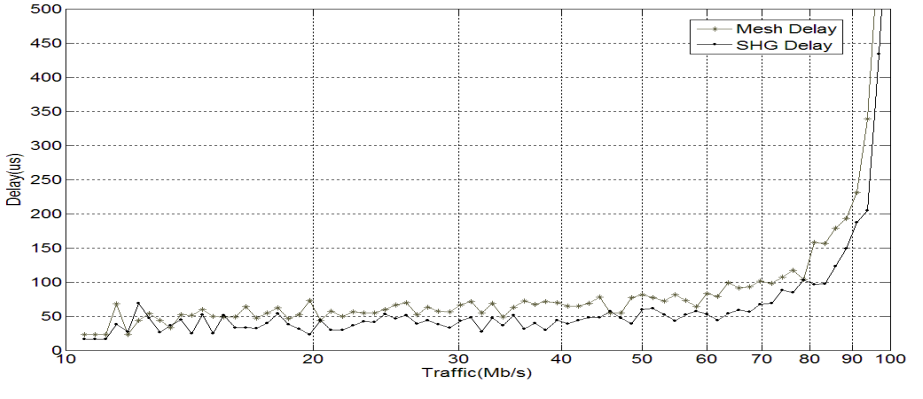


Fig. 7. Packet delay vs. traffic load of a random CBR source in a local traffic network (local traffic model)

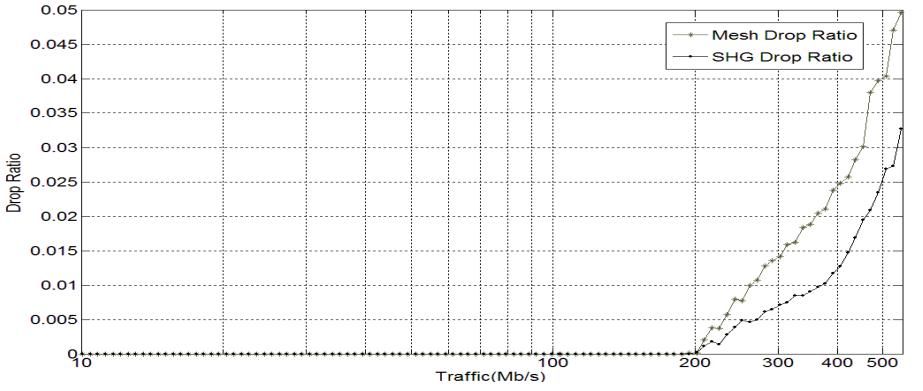


Fig. 8. Drop ratio vs. traffic load of exponential sources in a local traffic network (local traffic model)

Unlike the random traffic model, SHG shows better results for drop ratio in local traffic simulations. The reason is that in SHG there exist neighbor nodes with only one switch between but in Mesh the minimum number of switches between two resources is two. In other words SHG shows better locality than Mesh. Exactly the same reason stands for the bigger gap in delay of SHG and Mesh in higher loads.

In Fig.8 the drop ratio of SHG is much less than Mesh and the difference becomes more as the traffic load increases.

Fig.9 shows the delay of delivered packets for a local traffic model using exponential traffic sources.

These graphs shows that in lower traffic loads SHG performs better by a constant factor, and as the load increases the difference becomes more.

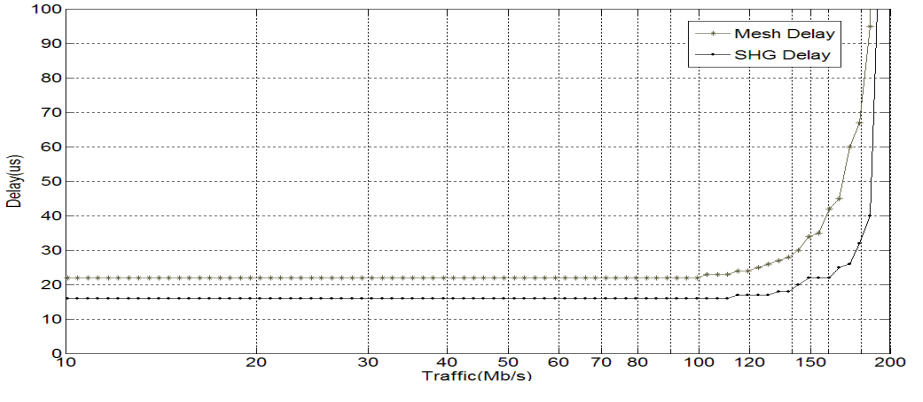


Fig. 9. Packet delay vs. traffic load of exponential sources in a local traffic network (random traffic model.)

6 Conclusion

This paper introduced HG as a flexible topology, specially designed for on chip networks. The flexibility of HG makes it suitable for various applications of SoC. HG provides a network infrastructure that can be easily altered to suite different applications. As an example SHG was introduced and benchmarked with a common used architecture (Mesh). The simulations proved that while SHG is easier to implement it outperforms Mesh in many aspects.

7 Future Work

The performance of HG depends on the appropriate choice of different topological parameters and many other factors such as switch buffer size, bandwidth and etc. The best approach is to find suitable values for these parameters for specific applications. These problems can be addressed in future work.

Furthermore, there are uncovered issues such as congestion and reliability control protocols, routing protocols and many other issues that are common among a very wide range of applications.

References

1. ITRS. International technology roadmap for semiconductors - 2004 edition, http://public.itrs.net/
2. L. Benini and G. De Micheli, Networks on Chip: A New SoC paradigm, IEEE Computer, January 2002(Vol. 35, No. 1), pp. 70-78.
3. JD Day, H Zimmermann, "The OSI reference model", Proceedings IEEE, 1334-1340, December 1983.
4. Shashi Kumar, et al., "A Network on Chip Architecture and Design Methodology", IEEE Computer Society Annual Symposium on VLSI, Pittsburgh, Pennsylvania, USA, April 2002.
5. Yi-Ran Sun, Shashi Kumar, Axel Jantsch, "Simulation and Evaluation for a Network on Chip Architecture Using Ns-2", Proceedings of 20th NORCHIP conference, Copenhagen, November 2002.
6. Vu-Duc Ngo and Hae-Wook Choi "On Chip Network: Topology design and evaluation using NS2".
7. Daniel Wiklund, Sumant Sathe, and Dake Liu, "Network on chip simulations for benchmarking", Proceedings of the International workshop on SoC for real-time applications, Banff, Canada, July 2004.
8. Nikolay Kavaldjiev, Gerard J. M. Smit, Pierre G. Jansen "A Virtual Channel Router for On-chip Networks", Proceedings of IEEE International SOC Conference, Santa ClaraSep, California, 2004
9. The network simulator - NS-2, http://www.isi.edu/nsnam/ns/

RISC/DSP Dual Core Wireless SoC Processor Focused on Multimedia Applications

Hyo-Joong Suh[1] and Jeongmin Kim[2]

[1] School of Computer Science and Information Engineering,
The Catholic University of Korea,
San 43-1 Yeoggog 2 Dong, Wonmi Gu, Bucheon, Gyeonggido, 420-743 Korea
hjsuh@catholic.ac.kr
[2] GCT Semiconductor, Inc.,
2121 Ringwood Avenue San Jose CA 95131

Abstract. The key technology of the mobile multimedia application is System-On-Chip processor that integrates the major function unit with low power consumption using a small battery. The wireless communication is practical technology, which makes comfortable communications between devices, such as IEEE 802.11a/b/g and Bluetooth. By the modern streaming technology with compression, like mpeg, the wireless communication triggers high quality media service without wires.

The wireless communication spend a quite large portion of energy of mobile device if data flows continuously, hence a delicate energy control may needed to fulfill a saving of the limited energy. Due to these issues, integrating the wireless function units into the high performance embedded processor supports a seamless control of power consumption of wireless part, and the communication data flows are sealed in the SoC chip, which reduces inter-chip communiaction energy between a main processor and a wireless media access controller.

We focused the high performance wireless multimedia applications are suffered by the energy consumption, so we alleviated the power issue by integrating of IEEE 802.11a/b/g media access controller, modem, ADC/DAC, high performance DSP, and RISC core in a silicon.

We present a wireless capable SoC processor, GDM5104, that integrated the multiple wireless media access controller as well as CCK/ OFDM/ GFSK modem and ADC/DAC also. Furthermore, the processor integrates RISC and DSP independent cores with appropriate caches, and rich peripherals that sufficient to implement mobile multimedia applications, hence the processor exhibits wireless connectivity with low power consumption. The processor is fabricated in a 0.18um standard CMOS technology, and operates at 133MHz RISC, and 100MHz DSP, which provides full capability of wireless multimedia processing.

1 Introduction

The progress of sub-micron technology allows an integration of various function units on a single system-on-chip (SoC), and appearances of some sophisticated

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 321–330, 2005.
© IFIP International Federation for Information Processing 2005

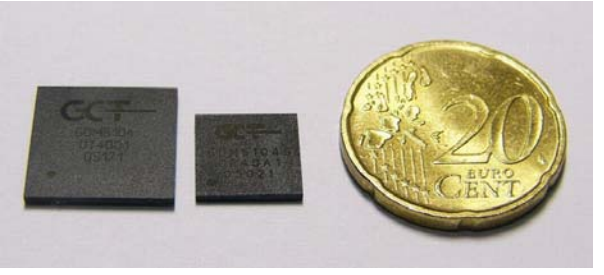


Fig. 1. GDM5104 embedded processor(256 pin, 169 pin fpBGA package)

multimedia processing algorithms induce a lot of multimedia applications that come onto the market. In terms of mobile applications, power energy is provided by primary or secondary batteries, which capacity is quite small to process a complex computation. Several energy saving methods are studied that include frequency control [1], voltage control [2], power control of peripheral block[3], and some power management techniques are implemented on the off-the-shelf embedded processors such as PXA270 by Intel[4].

Portability of a mobile application can be maximized if it communicates using wireless protocol. IEEE 802.11a/b/g [5] wireless standards are one of the rapidly imported technologies that applied on the mobile systems, however the wireless communications consume a lot of energy than wire communication standards such as USB and IEEE 1394. Hence the wireless mobile application is suffered by a shortage of energy, thus operation time is insufficient to play some multimedia titles.

Another issue arises from digital multimedia processing that consumes a lot of processor performance. Many multimedia processing algorithms are inefficient if it operates on the common reduced instruction set computer (RISC) style microprocessor, and if the multimedia algorithm processed on the RISC processor, it consumes lots of energy than a special unit for processing of the algorithm. By the viewpoint of energy consumption and performance, digital signal processing (DSP) processor is suitable to process a most multimedia algorithms, while the RISC processor performs better a general portion of the application code like user interface and TCP/IP protocol [6]. One example of the voice over wireless LAN that utilizes the wireless technology on mobile handset using voice over internet protocol. In this case, DSP processor supports to process speech vocoder, line/acoustic echo cancellation, DTMF, and CPT also. Parallel with the DSP processor, RISC processor supports the operating system, TCP/UDP/IP, RTP, device control, and other user interface parts.

By integrating of IEEE802.11a/b/g, RISC processor, DSP processor, USB2.0, NAND flash, IEEE802.3 MII interface, ATA, Bluetooth [7], and other peripherals on a chip, GDM5104, which presents a capability of powerful and energy-efficient operation in case of wireless mobile applications.

Next section presents the detailed processor internals of chip and low power features. Section 3 describes operating system, software environment, and debug

interface of the processor, and Section 4 shows early-stage performance evaluation. Finally, concluding remarks appear in Section 5.

2 GDM5104 Embedded Processor

The GDM5104 is a successor of GDM5302 [8], and these processors are wireless multimedia chips, which are under development. The advanced points of GDM5104 compared to the prior processor are independent structure of RISC and DSP dual processor cores and integration of wireless MAC/modem. Core of the prior GDM5302 processor was RISC/DSP hybrid structure that the core operates DSP mode or RISC mode exclusively, while the successor, GDM5104, has independent cores with independent caches respectively.

Figure 2 shows the dual cores and connected peripherals that exhibit lots of wireless and wire connectivity standards. Internal memory bus (IMB) is the high bandwidth internal bus that connects two cores and high-speed SDRAM, USB2.0, and DMA unit. Peripherals, DMA, and processor cores are connected via AMBA open standard bus [9] with enhanced features. There are two AHB buses of internal and external and APB bus for register access of peripherals. Each peripheral has two AHB bus masters for both AHB buses and one APB slave that will be used to access internal registers or command mapped coprocessor memory. XMEM (X-memory) and YMEM (Y-memory) are AHB slaves to internal AHB, and external memories are AHB slaves to external AHB. Therefore, internal AHB bus is used for the access between internal memories and AHB masters, and external AHB bus is used for the access between external memory and AHB masters. The enhanced features are multi-clock supporting and multiple byte write operations which provides independent bandwidth control on each block to control an energy consumption of each block.

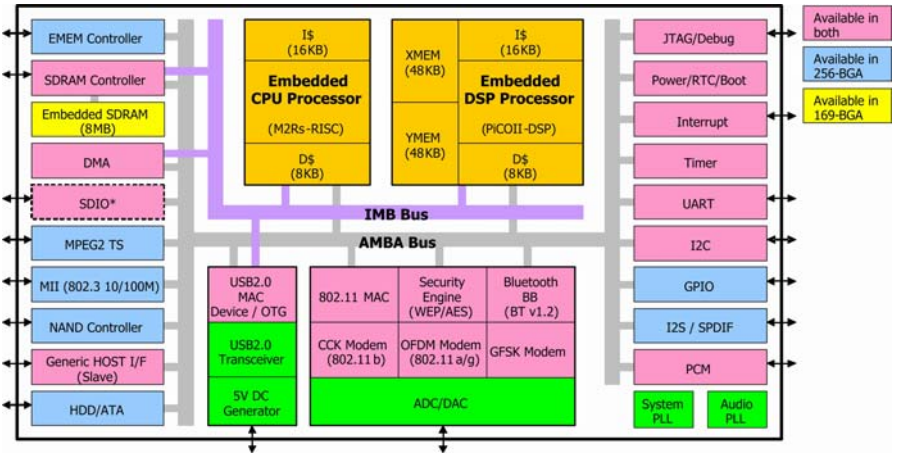


Fig. 2. Internal blocks of GDM5104 processor

2.1 Internal Bus Architecture

Detailed bus architecture of the processor is shown in Figure 3. The memory access priorities of each masters are decided as round robin in AHB arbiter. Since external memory is only connected to external AHB bus, its access priority is dependent on the AHB arbiter. However, internal memory such as XMEM and YMEM is not only connected to internal AHB bus, but also connected to internal CPU and DMA directly. Memory accesses from internal AHB bus have highest priority than the lower priority accesses from processors and DMA requests. AHB and APB bus operates at 88 or 80 MHz fixed frequency while each block is connected to bus through interface logics that can operate by ratios of 44 or 40 MHz reference clock. Speed of function blocks are tuned by the requirement of the bandwidth, thus they operate using multiple or fixed frequency by application requirements.

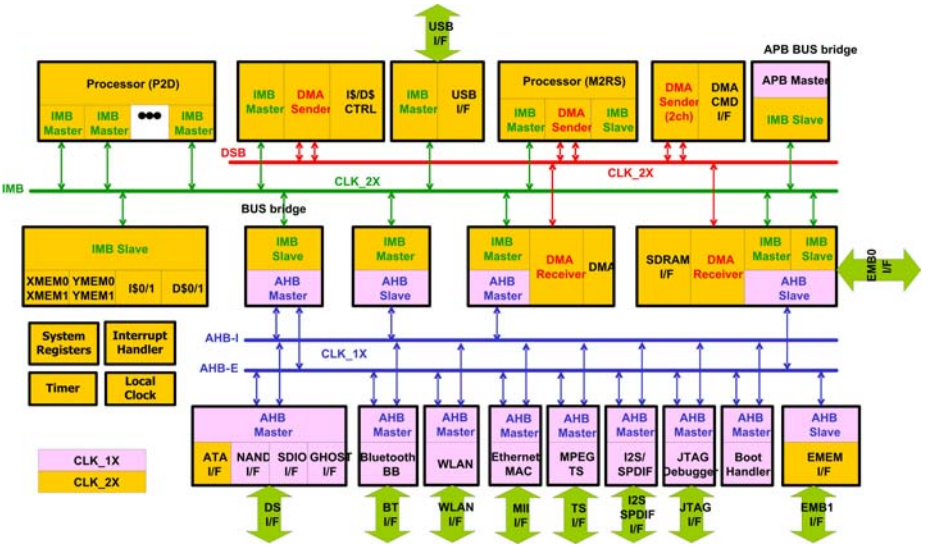


Fig. 3. System bus and operation clock of each block

2.2 DSP Core

The architecture of the PiCOII-DSP (P2D) core is RISC/DSP hybrid structure that has the capable of processing of DSP instructions as well as RISC instructions also. The P2D was the main core of its precedence processor, GDM5302. Like other conventional RISC, P2D adopts fixed length instructions format for simple decoding, 32 general purpose registers, and load-store architecture. Furthermore, P2D supports additional features to support DSP application effectively, P2D adopts an extra datapath for MAC-base operations used in the conventional DSP. In addition to MAC-based datapath, P2D includes several

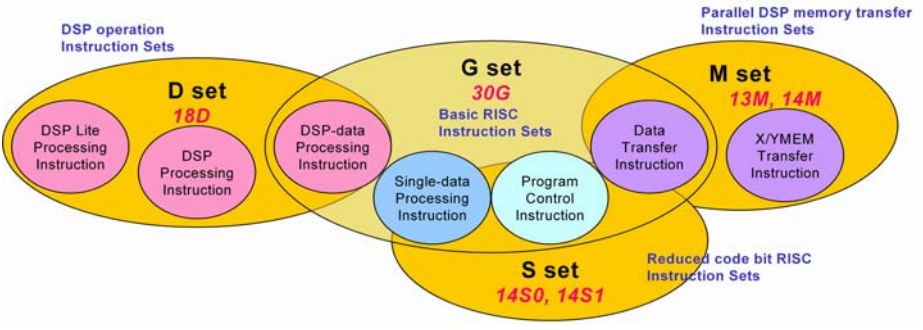


Fig. 4. Instruction sets classification of the P2D core

DSP operations such as saturation, scaling and round operations [10] in the instruction sets. Also, it supports Single Instruction Multiple Data (SIMD) features, which results in high performance in 16-bit speech applications [11]. The shift amount field of the special DSP mode registers automatically scales the source or destination registers. In addition to the data cache, P2D employs the parallel memory access operations of the internal data memories such as XMEM or YMEM. Thus P2D provides high memory bandwidth requirements in real-time multimedia applications. For XMEM or YMEM accesses, P2D supports memory accesses with special registers for address calculation. With the special address mode registers, P2D supports various addressing modes used in the conventional DSPs such as circular and bit-reverse addressing, which are used in many DSP applications. The features of P2D can be summarized as follows.

- RISC and DSP operation, single G set issue, dual S set issues, dual M set issues, both D set and M set operation simultaneously
- 32bit fixed instruction set
- Five stage pipeline : FE - DE - E0 - E1 - E2
- Interrupt disable during instruction execution within delayed slot
- 4read-2write register file
- 16bit/32bit DSP operation with one 48bit accumulator and 4 address registers
- 24bit x 24bit signed multiplication
- SIMD operations
- 1 delay slot for decode stage branch and execution stage branch and link
- 2 delay slot for execution stage branch
- 1 delay slot for load operation

P2D core has RISC and DSP instruction sets by its hybrid architecture, the instructions can be classified by four sets. Figure 4 describes detail of instruction sets and classifications.

G set Basic RISC instruction set that including integer scalar and DSP data processing with single memory transfer and control handling. Encoded as 30bit width.

S set Compact encoded width which provides a subset of G instruction set with reduced opcode and register field. Encoded as 14bit width.

D set Fixed 16bit or 32bit DSP operation including accumulation. Encoded as 18bit width with extending register field.

M set Specialized for internal scratch-pad memory transfers with several addressing mode and address registers. Encoded as with 13 or 14bit width dependent on the size of register field.

2.3 RISC Core

To optimize power consumption without sacrificing a performance that needed to operate a stream data manipulation with general network protocol and/or user interface processing, MiCROII-RISC (M2R) core is the embedded processor that used in mobile system that requires high-performance general operations along with the signal processing by the P2D. Instruction of the M2R core is the same as *G set* which is subset of the P2D instructions. It also has basic DSP data processing capability that accelerate simple computation in the protocol and display of user interface processing. The features of M2R can be summarized as follows.

- RISC with single 30G set issue
- 32bit fixed instruction set
- Six stage pipeline : FE - PD - DE - E0 - E1 - E2
- Interrupt disable during instruction execution within delayed slot
- 3read-1write register file
- 2 delay slot for decode stage branch and execution stage branch and link
- 3 delay slot for execution stage branch
- 2 delay slot for load operation

Each processor core adopts a programmable dynamic clock control that balances power consumption by the complexities of protocol tracking, user interface, and DSP processing. Clock speed of each core can be programmed in a real-time in order to match a requirement of target application. Each core also provides four states to control an energy, active state, sleep state, deep sleep state and power-off state. In active state, processor can change the processor clock between normal operation clock, a half of one, and a third of one. When a program execute *idle* instruction in the active state, processor pull down the processor clock without affecting the I/O device operation. Under the sleep state, whole system clock includes processor and I/O device disabled except real time clock (RTC). Wake-up will be occurred by the RTC time-out or external trigger signal since on-chip phase locked loop (PLL) is still working in order to fast response. An extreme power saving is possible if the processor enters to the Deep sleep state, which the whole clock stopped including PLL. Wake-up from the Deep sleep states takes more latency because it requires PLL boot-up time.

Each core has independent caches to reduce memory access latency. 16 Kbytes instruction and 8 Kbytes data caches on each RISC and DSP cores, and furthermore, the DSP core have scratch-pad memory such as 48 Kbytes XMEM and 48

Kbytes YMEM [12]. On-chip data cache can be configured as copy-through or copy-back mode dependent on the section indicator of address field. In cooperate with data cache, total 96 Kbytes internal scratch-pad memory is integrated enough to support hardware FIFO for each peripheral, which results in cost-effective and low power consumption systems. Whole memories can be byte and word accessible to save a valuable internal space as general-purpose data memory. Boot-up code of the processor can be placed in diverse devices such as NOR type flash, NAND type flash, and serial EEPROM, which reduce a total cost of system by minimizing a number of memory devices.

2.4 Wireless Connectivity

The GDM5104 medium access controller implements the IEEE 802.11 MAC sublayer using dedicated hardware and embedded firmware. MAC hardware implements real-time functions including data pump at transmit mode, validation check and filtering at receive mode, access protocol management, and DMA transferring of data between baseband and internal shared memory. Also, the most time critical frame such as ACK, CTS or response to CF-poll can be executed by MAC hardware itself. The embedded processor handles both Lower MAC and Upper MAC altogether as well as upper layer networking protocol such as TCP/IP. The lower MAC software handles the control of MAC hardware directly and its related lower layer of MAC protocols. It also handles the WEP hardware engine for encryption and decryption. The upper MAC software handles the BSS management, power state control, Beacon generation, data queue management, and host interface. The embedded processor runs low-layer firmware that manages the flow of command/data and upper-layer firmware to/from the application including the scan and join operations, and maintains transmit and receive queues, as well as a WEP key and engine control. Of course, the embedded CPU can control the time critical frame in the manual mode.

2.5 Peripheral Interface

The GDM5104 provides various peripheral interfaces such as off-chip memory interface, host interface, PCI/mini-PCI/CardBus, ATAPI, USB, UART, PCM, I2S, SPDIF, JTAG, Flash Memory/Card, and up-to 100-general purpose programmable I/O(GPIO) interfaces. The GDM5104 has two systems bus called IMB and AMBA. The IMB bus is specialized for high-speed communication between high bandwidth devices such as CPU, USB2.0, on-chip memory, and external SDRAM. Operation clock of the IMB bus is the same as processor clock to reduce the latency for high bandwidth data. Through the superior high performance bus, the embedded processor provides a sufficient bandwidth to execute the application that requires the large size of code and high throughput data movement. For the general peripheral data communication, each block is connected to AMBA AHB and APB bus also, which the AHB provides higher speed than the APB.

Table 1. Chain tools

tool	description
vincent-elf-gcc	Vincent C compiler
vincent-elf-g++	Vincent C compiler
vincent-elf-as	Vincent assembler
vincent-elf-ld	Vincent linker
vincent-elf-ar	Vincent librarian
vincent-gdb	Vincent source-level debugger
vincent-binutils	Vincent binutils
elf2flt	used to create a flat format output executable binary file
genromfs	used to create a file image for flash from a directory

3 Software Environment

3.1 Operating Systems and Tool Chain

Linux is one of the prominent operating system in the embedded world. Open source kernel and rich application can be migrated into the embedded application easily, and its powerful network protocols and applications are providing the network accessibility in the embedded system. We select the uClinux operating system on the GDM5104, which is one of the variations of Linux for embedded systems. For applications that require preemptive kernel, eCos[13] on GDM5104 is under development also. Vincent Software Development Toolkits (VSDT) is a development package that includes the basic tool chain and programming software to support the RISC and DSP processors. Sparc-Solaris, x86-Linux, and x86-cygwin platforms are supported to develop applications using VSDT.

3.2 Debug Support

The P2D and M2R cores have on-chip hardware debugger to support run-time debugging, and the on-chip debugger cooperates with external off-chip debuggers, GCoH and GNU debugger. JTAG, IPC, and PCI interfaces can be used as a communication channel between on-chip and off-chip debuggers. Figure 5 shows the debugging environment operates on run-time. Off-chip debugger controls on-chip debugger through the debug command register (DEBUG_CMD). Five types of commands are supported by DEBUG_CMD: SET, RUN, INTERRUPT GENERATION, RESOURCE ACCESS, and BIST commands. Special bit fields are prepared in the cores that generate internal exception for the purpose of hardware debugging only. By the value of each bit, the corresponding exception will be generated internally.

By control of the external off-chip debugger, the processor cores can be configured as one of the three processor modes, which are normal run, debug wait, and debug run. Under normal run mode, the on-chip debugger only provides limited commands related with memory and non-conflict resources with processor

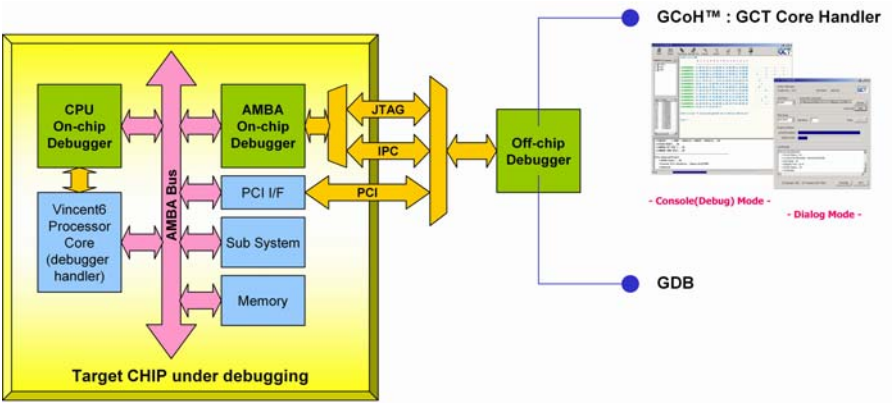


Fig. 5. On-chip and off-chip debug interface

core. Debug wait mode provides holding of execution that cooperated with the off-chip debugger. Under debug wait mode, the pipeline of the processor core is stopped but other sub-systems, such as peripheral devices, are working except timer. The timer can stop or work according to the specific configuration. Under the debug run mode, processor executes operations but the execution will be stopped at the preset counts of instructions or predefined breakpoint position by the configuration. The boot-up state of processor mode is defined by external chip interface signals, and off-chip debugger can change each mode anytime.

4 Performance Evaluation

By the M2R and P2D dual core architecture, it offers unique advantages in performance and power consumption for wireless multimedia systems. The M2R is well suited for handling protocol, user interface code, OS, and applications. The P2D, on the other hand, is better suited for the real-time signal processing applications commonly used in multimedia codec. In this paper, we present several audio codec performances using P2D core as shown in Table 2. In terms of optimization, quality of the source code is in early-stage to conclude the performance of P2D, so we expect the performance will be improved by the profiling of the code, further detailed performance will be released within near future.

Table 2. Performance of audio processing codecs using P2D (Mcycles/sec)

AMR		G.729AB		G.723.1 (6.3Kbps)		G.169
encoder	decoder	encoder	decoder	encoder	decoder	
25.5	4.4	21.5	3.9	29.9	2.3	15.7

5 Conclusion

Advent of the low power embedded processor pushes the digital appliances to an outdoors. The wireless technology enlarges a capability of the mobile application by its excellence of connectivity compared to wire communications. One of the most laborious obstacles is saving of the energy of mobile applications that operate using primary or secondary batteries. For the low power consumption and high computation/data-rate applications with the wireless connectivity, we integrate the high performance RISC and DSP cores with most wireless standards, IEEE802.11a/b/g and Bluetooth, on single silicon. By the seamless control of function units with high performance buses, every complicated application that require various peripheral operations, complex computations, and high-speed wireless communications can be offered using minimal power.

References

1. G. Semeraro and G. Magklis, R. Balasubramonian, D. Albonesi and S., Dwarkadas and M. Scott: Dynamic frequency and voltage control for a multiple clock domain microarchitecture Intl. Symp. on Microarchitecture (MICRO) (2002) 356–367
2. A. Iyer, D. Marculescu: Power efficiency of voltage scaling in multiple clock Proc. Intl. Conf. CAD (2002) 379–386
3. E. Macii, M. Pedram and F. Somenzi: High-Level Power Modeling, Estimation and Optimization Proc. DAC (1997) 504–511
4. Intel PXA27x Processor Family Developer's Manual (2004)
5. Brian P. Crow, Indra Widjaja, Jeon Geun Kim, Prescott T. Sakai: IEEE 802.11 Wireless Local Area Networks IEEE Commun. Mag. Vol.35 N.9 (1997) 116–126.
6. K. Brandenburg: MP3 and AAC Explained K. Brandenburg. MP3 and AAC Explained 17th Intl. Conf. High Quality Audio Coding (1999)
7. J. C. Haartsen: The Bluetooth radio system IEEE Pers. Commun. Mag. (2000) 28–36
8. H-J. Suh, H.J. Lim, J. Kim: GDM5302: Wireless Integration of a Mobile Multimedia Processor 2005 Intl. Conf. Embedded Systems and Applications (2005) 182–187
9. AMBA Specification rev 2.0 ARM (1999)
10. K. Wires, M. Schulte and J. Stine: Variable-Correction Truncated Floating Point Multipliers 34th Intl. Asilomar Conf. Signals, Circuits and Systems (2000)1344–1348
11. S. Kanthak, K. Schutz and N. Ney: Using SIMD Instructions for Fast Likelihood Calculation in LVCSR Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (2000)
12. M. Kandemir and A. Choudhary: Compiler-directed scratch pad memory hierarchy design and management 39th ACM/IEEE Design Automation Conf. (2002) 690–695
13. eCos User Guide Red Hat, Inc., eCosCentric Ltd. (2003) available on http://ecos.sourceware.org/docs-latest/

An Accurate Architectural Simulator for ARM1136

Hyo-Joong Suh[1] and Sung Woo Chung[2,⋆]

[1] School of Computer Science and Information Engineering,
The Catholic University of Korea,
San 43-1 Yeoggog 2 Dong, Wonmi Gu, Bucheon, Gyeonggido, 420-743 Korea
hjsuh@catholic.ac.kr
http://esca.catholic.ac.kr
[2] Department of Computer Science,
University of Virginia,
Charlottesville VA 22901 USA
schung@cs.virginia.edu

Abstract. Cycle-accurate simulators are basic tools to evaluate performance improvements of computer architecture. Before confirming of the architecture improvements using cycle-accurate simulation, the simulator itself should be validated. However, off-the-shelf processors have been continuously improved, though the cycle-accurate simulators were not reflected the improved features. Simulation results show that the difference between the IPC (Instruction Per Cycle) of the modified model for ARM1136 (Sim-ARM1136) and the IPC of the original model for ARM7 (Sim-Outorder) is 19%, on average, which is large enough to mislead the impact of architecture improvements.

1 Introduction

Every architect try to improve the performance of microprocessors to achieve a high speed, low power consumption, and large bandwidth, by examining a real silicon takes a lot of resources, cost and time. Practically, however, we can not explore all architectural alternatives on the silicon, and most researchers, especially in academia, do not analyze the impact of their idea in the silicon level. Instead, they disbranch useless alternatives using architecture simulators [1][2][3][4][5]. According to detail of the simulation level, the architecture simulators can be classified into two classes : instruction-accurate and cycle-accurate. Instruction accurate simulators aim at instruction level performance evaluation, which does not includes the detailed cycle information, and the correctness is limited within the instruction level. Compared to the instruction accurate simulators, cycle accurate simulators are focused on the lower level that requires the performance evaluation as well as correctness check.

About a year ago, we had been trying to deploy an existing processor simulator for architecture explorations. However, insufficient degree of details made

⋆ Corresponding Author.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 331–339, 2005.
© IFIP International Federation for Information Processing 2005

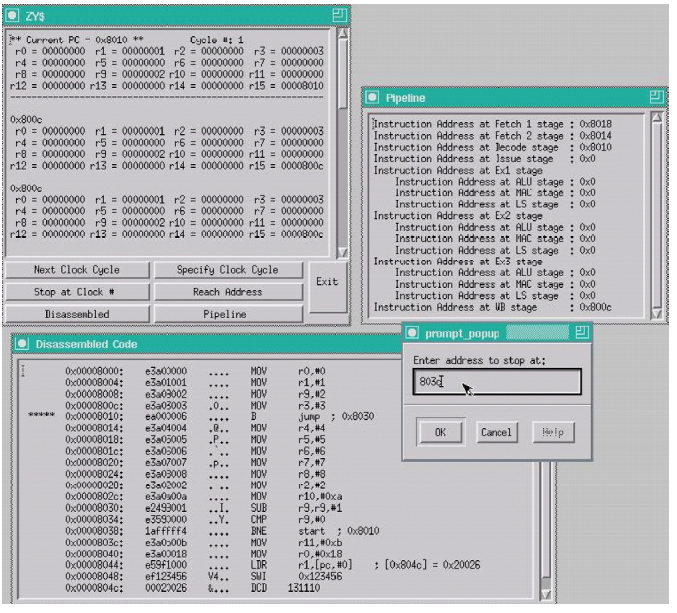


Fig. 1. Sim-ARM1136 GUI

us hesitate to utilize the existing simulators as they are. We worried about the abstraction errors that may draw wrong conclusions when evaluating new techniques [6].

Cycle-accurate simulators are powerful tools to evaluate a performance sensitive architecture alteration with time and cost efficiency, while these tools must be validated against the processor architecture. Usually, cycle-accurate simulators tend to trade-off between exactness and simulation speed, and moreover, many improved features are added on off-the-shelf microprocessors, which weaken a rationale of outdated cycle-accurate simulator as an architecture evaluation tool. For example, the cycle-accurate simulator (called *Sim-Outorder*) from Simplescalar-ARM [5] was shown to be cycle-accurate by comparing the ARM 7 [7] silicon [8]. However, ARM 1136 [9] has a lot of different features from ARM 7 : number of pipeline stages, organization of execution units, write buffers, and so on. Nevertheless, many researchers use existing simulators not even modifying pipeline structure and functions in each pipeline, when they evaluate their embedded processor designs. In this paper, we will discuss the inaccuracy that arises from the use of an existing cycle-accurate simulator without modifications, and compare with modified simulator that reflects many architecture updates.

To discuss this issue, we modified Sim-Outorder from Simplescalar[10] to reflect the off-the-shelf processor (ARM1136) [9] features. The modified model is called *Sim-ARM1136* in this paper. We selected Sim-Outorder from Simplescalar to compare with, because it is known as one of the most popular cycle-accurate simulators for the computer architecture research. To validate Sim-ARM1136,

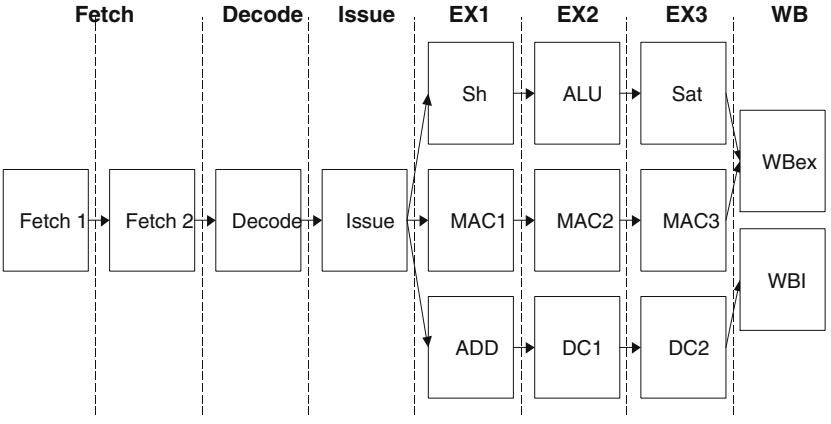


Fig. 2. Overall Pipeline Structure of Sim-ARM1136

we compare Sim-ARM1136 outputs with the RTL (register transfer level) simulation results of ARM1136. For convenience of operation, graphic interface was prepared on Sim-ARM1136, shown in Figure 1, and the user interface provided cycle-by-cycle validation against ARM1136 RTL by running some test programs.

In the next section, we describe Sim-ARM1136, focusing on the difference from Sim-Outorder. Section 3 describes the methodology that we used in this study. Section 4 shows the simulation results, and Section 5 concludes this paper.

2 Sim-ARM1136 Enhancements

In this section, we briefly describe Sim-ARM1136 enhancements compare with Sim-Outorder based on Simplescalar. Following features are the enhancement of Sim-ARM1136, and ARM1136 naturally.

– Pipeline stage modification

Sim-Outorder has 6-stage pipeline : fetch, decode, issue, execution, writeback, and commit. While Sim-ARM1136 is composed of 8-stage pipeline : fetch1, fetch2, decode, issue, execution1, execution2, execution3, and writeback as shown in Figure 2. Fetch and branch prediction consumes two cycles, resulting in two fetch stages, and execution is divided into three cycles.

In Sim-Outorder, one execution stage plays a role of multiple execution stages by setting latency and allowing multiple instructions in an execution unit. However, this structure can not support data forwarding in the execution stage. In Sim-ARM1136, we explicitly divide one execution stage into three execution stages for three kinds of execution units : Integer pipeline, MAC (Multiply and Accumulate) pipeline, and LS (Load/Store) pipeline.

Integer pipeline consists of Shift, ALU, and Saturation. If an instruction includes an operand to be shifted, the operand is shifted in the Shift stage. The

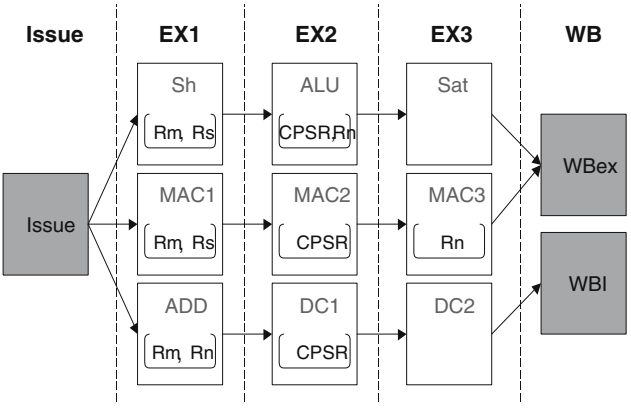


Fig. 3. Necessary Resolution of Data Dependency

ALU stage calculates the operands and the Saturation stage saturates the output from the ALU stage, if necessary. MAC pipeline consists of MAC1, MAC2, and MAC3. In the MAC1 stage and the MAC2 stage, operands are multiplied. In the MAC3 stage, the output from the previous stage is accumulated. The LS pipeline has ADD, DC1, and DC2. In the ADD stage, load/store address is calculated. If a shift operation is required to calculate the address in a load/store instruction, the shift operation in the Shift stage of the Integer pipeline should precede load/store instruction. In other words, if a shift operation is necessary for a load/store, the load/store instruction is implicitly split into two instructions. One goes through the Integer pipeline and the other goes through the LS pipeline. The data cache is accessed in the DC1 and DC2 stages.

The commit stage is merged into the writeback stage. We also modify the branch misprediction resolution mechanism to be appropriate for the modified pipeline structure.

– Interlock and data forwarding

In Sim-Outorder, instructions are issued after all data dependencies are resolved, while in Sim-ARM1136, instructions are issued regardless of data dependencies. In Sim-ARM1136, data dependencies are checked in every execution stage. If the required data dependency is not resolved, the instruction can not proceed until complete the resolution of the data dependency. Figure 3 shows the resolution of dependencies to proceed to the next stage. The dependency by the CPSR (Current Program Status Register) is also checked to match the condition code flags.

– Hit under miss

When Sim-Outorder is set to run in in-order, multiple data cache misses can go out to fetch data from memory by absence of second level cache in ARM1136. In case of a cache miss occurs, subsequent data cache access can fetch data only if there is a free entry in load store queue. In Sim-ARM1136, only one data

cache miss can fetch data from memory. If there is more than one cache misses, pipeline stalled until there remains only one cache miss.

– Memory interface

In modern processors, there is a write buffer between a first level cache and memory, whose purpose is to decouple stores from the pipeline. In Sim-Outorder, the entry of the write buffer is not limited. To reflect ARM1136 features, Sim-ARM1136 has an 8-entry write buffer, which may affect performance much. CWF (Critical Word First) is also modeled in Sim-ARM1136.

We also implemented 64-bit wide internal bus between the pipeline and the first level cache, which support two-word transfer at a time in case of LDM (LoaD Multiple) or STM (STore Multiple) operations.

– Static branch prediction

In Sim-Outorder, there are only dynamic branch predictions, while in Sim-ARM1136, static branch prediction as well as dynamic branch prediction is modeled also. ARM1136 uses bimodal prediction for dynamic branch prediction, and forward taken/backward untaken for static branch prediction. In the first fetch stage, dynamic prediction is performed and BTAC (Branch Target Address Cache) is accessed that is similar to BTB (Branch Target Buffer). In the second fetch stage, static prediction is performed if a miss occurs in the BTAC.

– Other miscellaneous enhancements

In the cache model of Sim-Outorder, there is no option for turn off the write allocation policy, while in ARM1136, a cache line for a read operation should be allocated in the cache.

MAC pipeline can not process MAC instruction for every cycle. At least one cycle stall is required between adjacent MAC instructions, and if the former MAC instruction is 64-bit operation, two cycle stalls is required. Data dependency and forwarding of the MAC instructions were implemented also, but we think the model of MAC instructions need to be detailed in Sim-ARM1136.

3 Methodology

We built the many enhancement of ARM1136 in Sim-ARM1136 as described, we discuss a validation process of Sim-ARM1136 and compare the accuracy of Sim-ARM1136 to that of Sim-Outorder in this section.

1). Simple test programs are made to verify the latencies and enhanced features in ARM1136 TRM[9] with Sim-ARM1136. These program includes the validation of ALU execution, MAC execution, interlock forwarding, hit under miss, and branch prediction/recovery.
2). Execution of Dhrystone (1-iteration) using ARM1136 RTL, Sim-ARM1136 and Sim-Outorder.
3). We assume the enhanced features of simulator are evaluated and stabilized if it provides similar improvement/degradation across

Table 1. Processor Parameters

Parameter	Value
Issue Policy	In-order
Branch Predictor	Bimodal, 128-entry
BTB (Branch Target Buffer)	128-entry
Instruction Cache	16KB, 32B line size, 4-way set associative
Data Cache	16KB, 32B line size, 4-way set associative
Cache Latency	1 cycle
Memory Latency	8 (first chunk), 12 (second), 16 (third), 20 (fourth)

other validated environments. In order to validation the stability of the simulators, we repeat procedure 2) by turning off the branch predictor. Turning off the branch predictor in ARM1136 RTL simulation can be done, by changing a status register value.

4). Procedure 2) and 3) are repeatedly done on Dhrystone (100-iteration)

5). Six applications are selected from the EEMBC benchmark suite, and compare the applications on ARM1136 RTL and Sim-ARM1136 with 1-iteration.

6). We compare the IPC of Sim-ARM1136 to that of Sim-Outorder on all the applications with default number of iterations by running the EEMBC benchmark suite. We could not run the applications on ARM1136 RTL, due to the running default iterations in the RTL simulation requires excessively long time.

Other processor parameters are complying with the ARM1136 features that detailed in Table 1. Please note that Sim-Outorder only accepts ARM7 binaries. For an appropriate comparison, we use ARM7 binaries compiled by the RVCT (RealView Compilation Tools) [12] as an input for all three simulators (RTL simulation, Sim-Outorder, and Sim-ARM1136).

4 Simulation Results

4.1 IPC Comparison: ARM1136 vs. Sim-Outorder vs. Sim-ARM1136 (Dhrystone)

When running Dhrystone 1-iteration, the IPC (Instruction Per Cycle) from the ARM1136 RTL simulation is 0.19 as shown in Figure 4. In this case, the IPC from Sim-Outorder and that from Sim-ARM1136 is 0.22 and 0.20, respectively. All the results are similar. When simulating without the branch predictor, the relative performance loss in Sim-ARM1136 is as close as that in ARM1136. However, the performance loss in Sim-Outorder is as much as 24%. Different from Sim-ARM1136, Sim-Outorder can issue multiple outstanding data cache miss requests to memory. However, without the branch predictor, the effect of multiple outstanding data miss requests decreases.

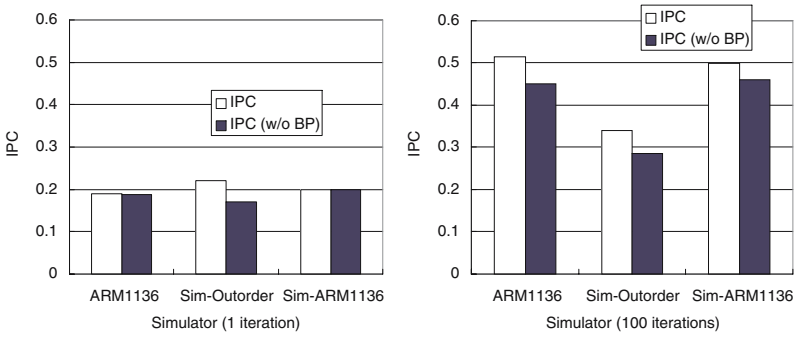


Fig. 4. IPC Comparison (Dhrystone)

Table 2. IPC Comparison between ARM1136 and Sim-ARM1136 (EEMBC, 1-iteration)

Applications	IPC (ARM1136)	IPC (Sim-Outorder)	Difference (%)	IPC (Sim-ARM1136)	Difference (%)
autcor00	0.57	0.73	-28.07	0.62	-8.77
bitmnp01	0.71	0.75	-5.63	0.71	0.86
bitmnp816	0.60	0.63	-5.00	0.55	8.33
memacc816	0.32	0.35	-9.38	0.35	-9.38
pntrch816	0.53	0.48	9.43	0.51	3.77
rspeed816	0.19	0.22	-15.79	0.25	-31.58
Mean	0.48	0.60	12.22	0.58	10.45

When running Dhrystone 100-iteration, the results from ARM1136 and Sim-ARM1136 shows less than 3% difference. Sim-Outorder has 36% lower IPC, compared to the ARM1136. There are two reasons for that. First, running more iterations reduce data cache miss rate, leading to decrease of the multiple data cache miss effect. Second, data dependency prevents instructions from being issued in Sim-Outorder, where only dependency-resolved instructions can be issued.

4.2 IPC Comparison: ARM1136 vs. Sim-Outorder vs. Sim-ARM1136 (EEMBC, 1-Teration)

Table 2 shows the accuracy of Sim-Outorder and Sim-ARM1136 to that of Sim-Outorder by running EEMBC benchmark suite [11]. The average difference is 12.22% and 10.45% in Sim-Outorder and in Sim-ARM1136, respectively. Main source of differences may be caused by not modeled features (more detailed MAC pipeline, return address stack, branch folding and so on).

Since the first iteration incurs much more data cache misses that account for large portion of execution time, the IPC of Sim-Outorder is very close to

that of Sim-ARM1136. In the next Subsection, we show the difference between Sim-ARM1136 and Sim-Outorder by running the default number of iterations of all applications in the EEMBC benchmark suite.

4.3 IPC Comparison: Sim-ARM1136 vs. Sim-Outorder (EEMBC, Default-Iteration)

As shown in Figure 5, the simulation results show that Sim-Outorder has higher IPC than Sim-ARM1136 in most applications. These results are reverse, compared to the results in Section 4.1. The reason is that EEMBC benchmark suite incurs much more data cache misses than Dhrystone, which results in performance gain from multiple outstanding cache misses in Sim-Outorder. Though Sim-ARM1136 allows the instructions whose data dependencies are not resolved, this effect is compensated by hit under miss feature (explained in Section 2) that allows only one outstanding data cache miss. The average IPCs for Sim-ARM1136 and for Sim-Outorder are 0.55 and 0.63, respectively. The average difference is as much as 19.0%. Moreover, the increased IPC in Sim-Outorder is not proportional among the applications. Therefore, the amount of performance improvement or degradation by researcher's new idea would be different between in Sim-ARM1136 and in Sim-Outorder.

Though Sim-ARM1136 shows lower IPC in most applications, Sim-ARM1136 shows higher IPC in some applications that are expected to have a lot of data forwarding and utilize the 64-bit wide internal bus. Especially, in puwmod816

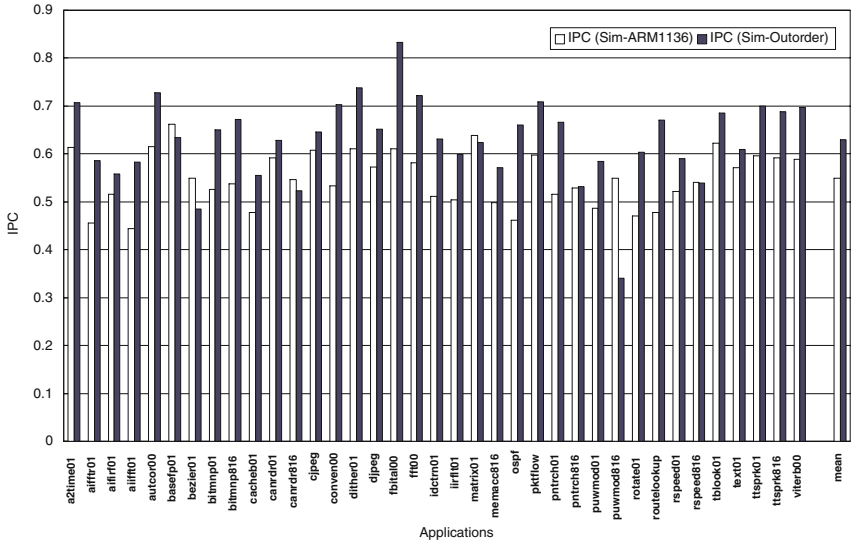


Fig. 5. IPC Comparison between Sim-ARM1136 and Sim-Outorder (EEMBC,default-iteration)

that has much more LDM/STM, the IPC difference is as much as 38%. Since LD/ST generates two operations (shift operation for address calculation and practical LD/ST), it seriously increases data dependencies. In this case, data dependency can be resolved by data forwarding, resulting in relative performance gain in Sim-ARM1136.

5 Conclusions

Simulators are essential tools for most researchers to analyze their valuable idea without costly silicon implementation. However, state-of-the art embedded processors have many architectural improvements than the simple simulators. By the absence of improved features of embedded processors on simulators, if researchers put ideas in the simulators without reflecting of the improvements, simulation results can be distorted by impacts of added features. Thus, this might unintentionally conclude valuable idea as useless.

In this paper, we validated Sim-ARM1136, which simulates the processor parameters of ARM1136 as well as the extended features. We also presented the difference between Sim-ARM1136 and Sim-Outorder where only processor parameters are changed without modifying pipeline features. The simulation results show the difference that might mislead to incorrect performance evaluation. Now, Sim-ARM1136 is adopted by Samsung electronics to evaluate of high-performance embedded processors.

References

1. V. Pai, P. Ranganathan, and S. Adve: RSim: A Simulator for Shared Memory Multiprocessor and Uniprocessor Systems that Exploit ILP In Proceedings of the 3rd Workshop on Computer Architecture Education (1997)
2. M. Rosenblum, S. Herrod, E. Witchel, and A. Gupta: Complete Computer Simulation : The SimOS Approach In IEEE Parallel and Distributed Technology (1995)
3. D.M. Tullsen, S.J. Eggers, and H.M. Levy: Simultaneous Multithreading: Maximizing On-Chip Parallelism In 22nd Annual International Symposium on Computer Architecture, June (1995)
4. Virtutech: Simics Simulator. available at http://www.virtutech.com.
5. Simplescalar LLC: Simplescalar 3.0 available at http://www.simplescalar.com.
6. Bryan Black and Jhon Paul Shen: Calibration of Microprocessor Performance Models Computer, 31 (5):41-49, May (1998)
7. ARM Corp.: ARM 7 Technical Reference Manual available at http://www.arm.com.
8. Simplescalar LLC: Simplescalar Tutorial available at http://www.simplescalar.com.
9. ARM Corp.: ARM 1136 Technical Reference Manual available at http://www.arm.com
10. Simplescalar LLC: Pre-release of Simplescalar 4.0, (2004)
11. The Embedded Microprocessor Benchmark Consortium: EEMBC Benchmark Suite available at http://www.eembc.org.
12. ARM Corp.: RealView Compilation Tools available at http://www.arm.com.

Modular Design Structure and High-Level Prototyping for Novel Embedded Processor Core

Ben A. Abderazek, Sotaro Kawata, Tsutomu Yoshinaga, and Masahiro Sowa

Graduate School of Information Systems,
The University of Electro-Communications,
1-5-1 Chofugaoka, Chofu-shi, 182-8585 Tokyo, Japan
ben@is.uec.ac.jp

Abstract. In this research work, we present a high-level prototyping of a new processor core based on Queue architecture as starting point for application-specific processor design exploration. Using modular design structure with control logic implemented as a set of communicating state machines, we show hardware emulation and optimizations results of a parallel queue proecssor architecture (QueueCore). We also show how to to fully exploit the capabilities of the designed QueueCore, while maintaining a common source base. From the evaluation results, we show that the QueueCore prototype fits on a single conventional FPGA device, thereby obviating the need to perform multi-chip partitioning which results in a loss of resource efficiency.

1 Introduction

In our previous work, we have researched about queue computing infrastructure as a starting point for cooperative design space exploration for general and embedded applications[1, 2]. The choice of this architecture was based on a number of factors, such as its suitability for embedded applications.

A parallel Queue processor architecture[2] exploits instruction-level parallelism without considerable effort and need for heavy run time data dependence analysis, resulting in a simple hardware organization when compared with conventional superscalar processors. These features allow the inclusion of a large number of functional units into a single chip, increasing parallelism exploitation.

The key idea of the Queue processor is the manupilation scheme of instruction operands and result. The Queue computing scheme adopted in this work stores intermediate results into a so called queue-registers (QREG). In this model, a given instruction implicitly reads its first operand from the head of the QREG, its second operand from a location explicitly addressed with an integer offset from the first operand location and stores the computed result into the tail of the QREG. An important feature of this scheme is that write after read false data dependency does not occur[1]. Furthermore, since there is no explicit referencing to the QREG, it is easy to add extra storage locations to the QREG when needed.

Since the operands and result addresses of a given static-instruction (compiler generated) are implicitly *computed* during run-time, an efficient and fast

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 340–349, 2005.
© IFIP International Federation for Information Processing 2005

hardware mechanism is imperatively needed for parallel execution. The proposed architecture, which is named QueueCore, implements a so named queue computation mechanism that calculates operands and result addresses for each instruction. The mechanism used for calculating these addresses will be described in the coming section. An important aspect of developing any new architecture is verification which usually requires complicated and lengthly software simulation of an emulated model. An event-based or cycle level simulation becomes increasingly inadequate to verify a significant execution trace for a given problem. These software-based simulation approaches are not capable of predicting all micro-architectural issues related to final physical design[3]. To this end, we consider prototyping-based emulation that substitutes real time hardware emulation for slow simulator-based execution. Prototyping-based design itself is not a new idea as several designers already adopted this method. However, what is important for us is to investigate how to describe the novel QueueCore architecture to achieve good synthesis results for FPGA implementation with sufficient performance to support realistic evaluation.

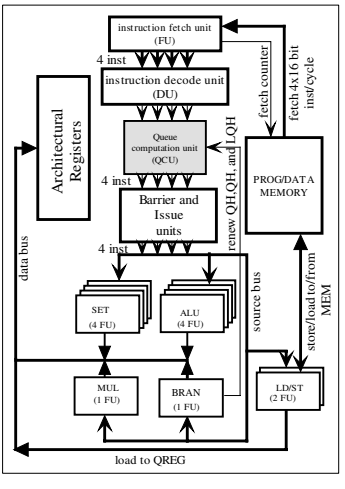


Fig. 1. QueueCore architecture block diagram. During RTL description, the core is broken into small and manageable modules using modular approach structure for easy verification, debugging and modification.

Using a hardware description language, we have created the Synthesizable model of the QueueCore for the integer subset of parallel Queue processor (PQP) architecture[1, 2]. A prototype implementation is produced by synthesizing the high-level model for the Stratix FPGA device[4, 6]. As results, we were able to evaluate relative circuit area and speed metrics for design alternatives.

2 QueueCore Architecture

The QueueCore block diagram is shown in Fig. 1. The system has six pipeline stages:

(1) *Fetch (FU)*: at each cycle the fetch counter is sent to PROG/DATA memory and eight bytes (four instructions) are fetched then inserted into the fetch buffer (FB).
(2) *Decode (DU)*: this unit reads eight bytes from the FB every cycle, decodes them, then inserts decoded information into the decode buffer. This stage also calculates the number of consumed (CNBR) and produced (PNBR) data for each instruction. The CNBR and PNBR are used by the next pipeline stage (queue computing stage) to calculate the sources and destination locations for each instruction.
(3) *Queue computation (QCU)*: calculates the first operand (*source*1) and destination addresses for each instruction. The mechanism used for calculating the *source*1 address in given in Fig. 2. The QCU unit keeps track on the current value of the QH and QT pointers. Four instructions arrive to the QCU unit each cycle. For the first instruction, the number of the consumed operands (CNBR) (8-bit field) is added to the current QH value (QH0) to find the address of the first operand (QHN) and the number of produced results (PNBR) (8-bit filed) is added to the current QT value (QT0) to find the result address (QT1) of the first instruction. The other three instruction's first operand and result addresses are calculated similarly as indicated in the lower part of Fig. 2. Notice that the calculation is performed sequentially.
(4) *Barrier:* inserts barrier flag for dependency resolutions.
(5) *Issue:* four instructions are issued for execution each cycle. In this stage, the

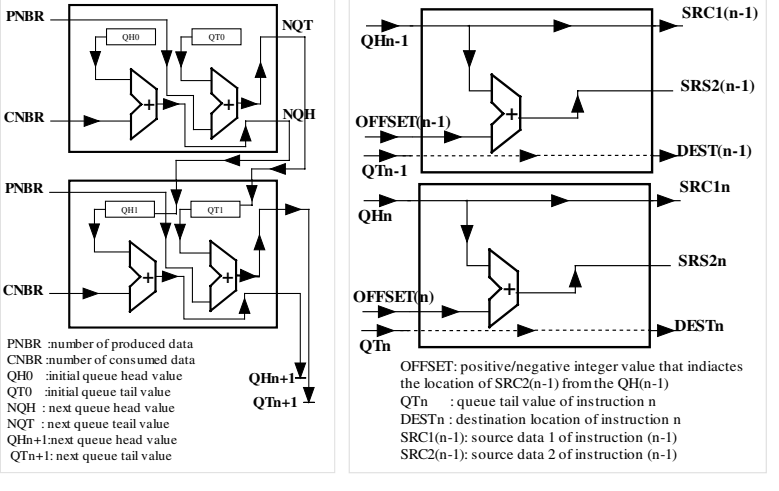


Fig. 2. Address calculation mechanism for *source*1, *destination* (left part of the figure), and source two calculation mechanism (right part of the figure)

second operand (*source*2) of a given instruction is first calculated by adding the address *source*1 to the displacement that comes with the instruction. The second operand address calculation could be earlier calculated in the QCU stage. However, for timing balance consideration, the *source*2 is calculated in the IS stage. The hardware mechanism used for calculating the second operand (*source*2) address is shown in the right part of Fig. 2.
An instruction is ready to be issued if its data operands and its corresponding functional unit are available.
6) *Execution (EXE)*: the QueueCore's execution unit computes the effective address for loads and stores, and executes ALU instructions. During the EXE stage, the machine also determines whether a conditional branch is taken and, if so, updates the program counter and send the QH and QT of branch instruction to the QCU unit.

3 QueueCore Implementation

To the best of our knowledge, this is the first hardware design of a Queue processor architecture. Therefore, there are no related works in the literature that give design issues or methodology related to hardware Queue processor.

To make the QueueCore design easy to debug, modify and adaptable, we decided to use a high-level description, which was also used by other system designers. We have developed the QueueCore system in Verilog HDL.

3.1 QueueCore System Pipeline Control

In many conventional processors, the control unit is centralized and controls all central processing core functions. This scheme introduces pipeline stalls, bubbles, etc. However, especially for pipelined architecture, this control unit is one of the most complex part of the design, even for processors with fixed functionality.

In this design, we have decided to break up the unstructured control unit to small, manageable units. Each unit is described in a separate HDL module. That is, instead of a centralized control unit, the control unit is integrated with the pipeline data path. Thus, each pipeline stage is mainly controlled by its own, simple control unit as illustrated in Fig. 3. In this scheme, each distributed state machine corresponds to exactly one pipeline stage, and this stage is controlled exclusively by its corresponding state machine. Overall flow control of the processor is implemented by cooperation of the control units in each stage based on communicating state machines. Each pipeline stage is connected to its immediate neighbours, and indicates whether it is able to supply or accept new instructions. Communication with adjacent pipeline stages is performed using two asynchronous signals, *AVAILABLE* and *PROCEED*. When a stage has finished processing, it asserts the *AVAILABLE* signal to indicate that data is available to the next pipeline stage. The next stage will, then, indicate whether it can proceed these data by using the *PROCEED* signal.

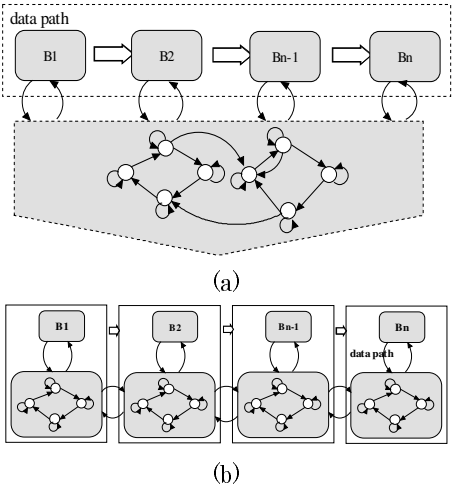


Fig. 3. Control unit: (a) In conventional Processor design, (2) in PQP processor design

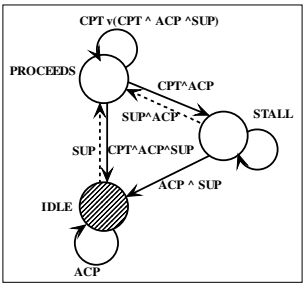


Fig. 4. Finite state machine transition for QueueCore pipeline synchronization. The following conditions are evaluated: next stage can accept data (ACP), previous pipeline stage can supply data (SUP), last cycle of computation (CPT).

Since all field necessary to find what actions are to be taken next are available in the pipeline stage(for example operation status ready bits and synchronization signals from adjacently stages), computing the next stage is simple. The state transition of a pipeline stage in the QueueCore is illustrated in Fig. 4. This basic state machine is extended to cover the operational requirements of each stage, by dividing the *PROCEED* state into substates as needed. An example is the implementation of the Queue computation stage, where *PROCEED* is divided into substates for reading initial addresses values, calculating next addresses values, and addresses fixup (when needed).

4 QueueCore Validation

Validation loosely refers in this paper to the process of determining that the designed processor is correct. It is performed at multiple abstraction levels by integrating several approaches performed in three phases and based on the same HDL that describe the processor core: (1)Functional verfication,(2)Rapid Prototyping, and (3)Gate Level Simulation. The QueueCore graphical validation tool (GVT) display the validated output ports when the source clock changes. A summary of the memory, QREG, the genral purpose registers (GPR) status and the instructions processing counters can be monitored with the GVT tool.

5 FPGA Synthesis of the QueueCore

In order to estimate the impact of the description style on the target FPGA efficiency, we have explored logic synthesis for FPGAs. The idea of this experiment was to optimize critical design parts for speed or resource optimizations. The optimizations of the Verilog HDL models for FPGAs can be arranged into two distinctive approaches: (1) Coding style and (2) Usage of special-purpose Devices on FPGA. In this work, our experiments and the results described are based on the Altera Stratix architecture [4]. We selected Stratix FPGA device because it has a good tradeoffs between routability and logic capacity. In addition it has an internal embedded memory that eliminates the need for external memory module and offers up to 10 Mbits of embedded memory through the TriMatrix TM memory feature. We also used Altera Quartus II professional edition [6] for simulation, placement and routing. Simulations were also performed with Cadence Verilog-XL tool [5]. The quality of the final QueueCore hardware and the resource usage of the target FPGA device depend very much on the description style used at a higher level. To account for this, the high-level HDL description has to be adapted to guide the synthesis tool to choose the appropriate implementation method for functionality. The coding style also affects the design of finite-state automata/machine (FSM) that are common is the QueueCore. A poor choice of codes will result in a state machine that uses too much logic, or is too slow, or both. Many advanced techniques have been developed for choosing an optimal state assignment. Typical such approaches use the minimum number of state bits or assume a two-level logic [7].

Fig. 5 compares the encoding tradeoffs for a simple FSM for the Altera STRATIX-EP1S FPGA and Structured ASIC library using three encoding techniques: *gray* code encoding, *binary* encoding, and *onehot* encoding of states.

When ASICs models are the target process, fully encoded representation such as binary or gray code encoding of states lead to space efficient design, wheres the one-hot encoding scheme consumes more resources. This is different for FPGA devices with a large number of flip-flops, where a decoded representation reduces decoding logic at the cost of typically underutilized flip-flops.

Since the resource distribution of the *onehot* encoded FSM maps well to the resource distribution of the target architecture, it is a very compact representation.

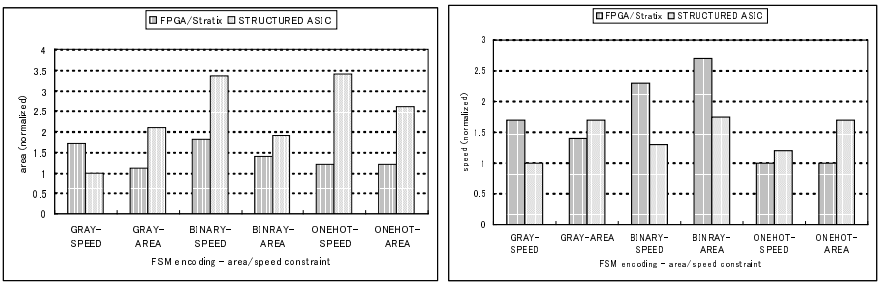


Fig. 5. Resource usage and timing for state machine synthesis using different encoding schemes

6 Modeling Style with Special FPGAs Circuits and Storage Structure

To compare different modeling styles and their implementation in hardware, we have considered the design for an adder module with several different algorithm:

DEDA uses of dedicated logic,
LGCA use only logic element,
DFTA use of default implementation

To compare these design styles, we have compiled these adders with and without target-specific module generators. Resource usage and timing of these implementations of a 32 bit adder are shown in Figures 6. The *DFTA* implementation was slightly better than *LGCA* and *DRDA* version, because the synthesis tool contains a powerfully library of module generators. Implementations not based on the fast arithmetic functionality of the FPGA have almost similar resource usage which is much higher than the technology-specific version, and their performance is slower by a factor of 1.1 to 1.4. Fig. 7 compares two different target

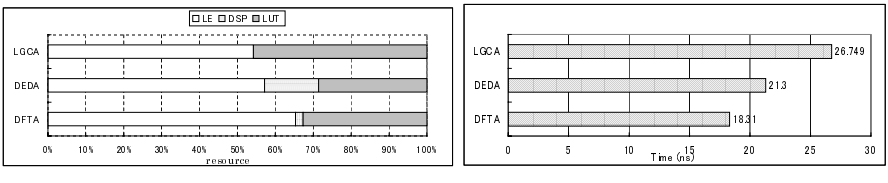


Fig. 6. Resource usage and timing for 32 bit adder

implantation for 256x33 QREG for various optimizations. Depending on the target implementations device, either logic elements (LEs) or total combinational functions (TC) are generated as storage elements. Implementations based on HardCopy device, which generates TCF functions give almost similar complexity

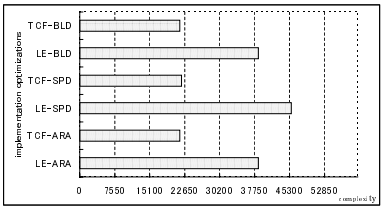


Fig. 7. Resource usage and timing for 256*33 bit QREG unit for different coding and optimization strategies

for the three used optimizations – area(ARA), speed (SPD) and balanced(BLD). For FPGA implementation, the complexity for SPD optimization is about 17% and 18% higher than that for ARA and BLD optimizations respectively.

7 Results and Discussions

Table 1 summarizes the synthesis results of QueueCore for FPGA Stratix and HardCopy devices. The complexity of each processor module as well as the whole processor core are given as the number of logic elements(LEs) for the Stratix FPGA device and as the TCF cell count for the HardCopy device (Stuructured ASIC). The design was optimised for BLD optimization guided by a properly implemented constraint table. Figure 9 shows the floorplan of the placed and routed QueueCore processor. The pipeline stages have been placed from the top right to the lower left corner of the target Stratix EP1S device. From the above experiment, we found that the processor consumes about 80.4% of the total logical elements of the target device.

Figure 8 shows the achievable throughput which can be obtained from this design on different execution platforms. For the hardware platforms, we show the processor frequency. For comparison purposes, the Verilog HDL simulator per-

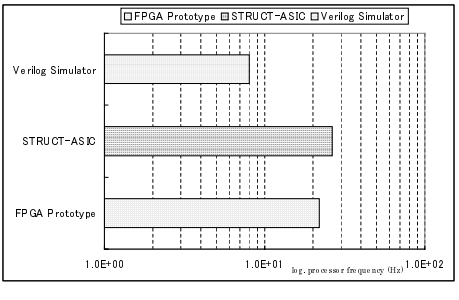


Fig. 8. Achievable frequency is the instruction throughput for hardware implementations of the QueueCore processor. Simulation speeds have been converted to a nominal frequency rating to facilitate comparison.

Table 1. Queue processor design results: modules complexity as LE (logic elements) and TCF (total combinational functions) when synthesised for FPGA (with Stratix device) and Structured ASIC (HardCopy II) families

Descriptions	Modules	LE	TCF
instruction fetch unit	IF	633	414
instruction decode unit	ID	2573	1564
queue compute unit	QCU	1949	1304
barrier queue unit	BQU	9450	4348
issue unit	IS	15476	7065
execution unit	EXE	15868	7241
queue-registers unit	QREG	38543	21590
memory access	MEM	5158	3936
control unit	CTR	171	152
Queue processor core	QueueCore	89821	47614

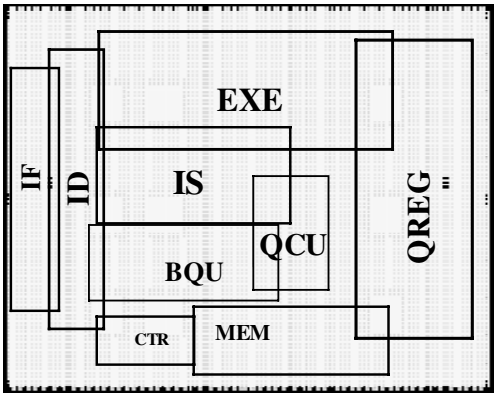


Fig. 9. Floorplan of the placed and routed processor core. The modules of the processor show considerable overlap as logic is mapped according to interconnect requirements.

formance has been converted to an artificial frequency rating by dividing the simulator throughput by a cycle count of 1 CPI. This chart shows the benefits which can be derived from direct hardware execution using a prototype when compared to processor simulation. The data used for this simulation are based on event-driven functional Verilog HDL simulation.

8 Concluding Remarks

A Queue (QueueCore) procerssor for Stratix FPGA was created using modular design structure in high-level prototyping system. The architecture is flexible, with library files controlling key definitions needed by various components. Recompilation can be done to make desired changes to pipeline information,

QREG size, memory size and various other architectural features. Each architectural components has been tested through extensive simulation, as well as actual implementation in the Stratix FPGA device.

We also described what constraints contribute to efficient FPGA implementation. This has been demonstrated by target processor core as a single high-density FPGA.

Evaluation results reveal that queue processor achieves a speed of about 22.5 and 25.5 MHZ for QREG16 (QREG size is 32*16 entries) and QREG264 (QREG size is 32*264 entries) respectively. We also found that the processor consumes about 80.4% of the total logical elements of the Stratix device. As a result, it fits on a single Stratix device with an internal embedded memory that eliminates the need for external memory module, thereby obviating the need to perform multi-chip partitioning which results in a loss of resource efficiency. Only, few clearly identified such as the Barrier and the QREG units need to be specially optimised at the HDL source level to achieve efficient resources usage.

References

1. M. Sowa , Ben A. Abderazek, T. Yoshinaga, "Parallel Queue Processor Architecture Based on Produced Order Computation Model", Int. Journal of Supercomputing, HPC, Vol. 32,No. 3, pp. 217-229, June 2005.
2. B. A. Abderazek, M. Arsenji, S. Shigeta, T. Yoshinaga, M. Sowa, Queue Processor for Novel Queue Computing Paradigm Based on Produced Order Scheme, Proc. of HPC, IEEE CS, 0-7695-2138-X/04, pp. 169-177, July 2004
3. M. Sheliga, and E. H. sha,Hardware/Software Co-design With the HMS Framework," Journal of VLSI Signal Processing Systems, Vol. 13, No. 1,pp. 37-56, 1996.
4. D. Lewis et al, The Stratix Logic and Routing Architecture, Proc FPGA-02, pp 12-20, 2002
5. Cadence Design Systems:http://www.cadence.com/
6. Altera Design Software: http://www.altera.com/
7. A.E.A Almaini, et al.: State Assignment of Finite State Machines using a Genetic Algorithm, IEE Proc. on Computers and Digital Techniques, pp. 279-286. 1995.

Pipelined Bidirectional Bus Architecture for Embedded Multimedia SoCs*

Gang-Hoon Seo, Won-Yong Jung, Seongsoo Lee, and Jae-Kyung Wee

School of Electronics Engineering, Soongsil University, 156-743, Korea
wjk@ssu.ac.kr

Abstract. This paper proposes novel high-performance bus architecture for memory-intensive embedded multimedia SoCs. It has a pipelined bidirectional bus for high speed and small area. It has two separate bus called system bus and memory bus, where memory-intensive IPs are connected to memory bus so not to degrade system bus performance. To avoid starvation of low-priority masters, the proposed bus exploits probability-based arbitration policy where the arbitration probability of each master is determined in proportion to its execution time. To increase transmission bandwidth, it also exploits bus partitioning where several masters often access their slaves concurrently without multilayer structure. The proposed bus is designed, implemented, verified, and evaluated in hardware level. Simulation results show that the proposed bus improves effective bandwidth by 2.8~3.6 times and communication latency by 3.1~4.7 times when compared to AMBA bus.

1 Introduction

Recently, advances of integrated circuit technology have enabled development of complex embedded systems into systems-on-chip (SoC) [1], where pre-designed IP cores are stitched together through various communication links. Design of high performance global communication architecture has become a key to successful embedded SoC designs. On-chip buses [2]-[6] are often used for on-chip communication between internal IP cores.

On-chip buses are classified into standard buses [2]-[5] and wrapper-based buses [6]. Standard on-chip buses specify its own protocol over wiring connections between IP cores. Any IP cores complying that protocol can be reused in other embedded SoCs using same bus type. However, interface logic should be redesigned when different bus type is used. Wrapper-based approaches use IP core interface protocol independent of physical bus protocol, and they use hardware wrappers to handle core-to-core communication. Hence, IP cores complying with the interface protocol can be integrated into embedded SoC, even if their physical bus types are different. However, complex wrapper architecture is required to reduce access latency. In practical applications, wrapper-based buses such as virtual component interface (VCI) and open core protocol (OCP) are widely used.

* This work was supported by grant No. R01-2005-000-10540-0 from the Basic Research Program of the Korea Science and Engineering Foundation.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 350 – 359, 2005.
© IFIP International Federation for Information Processing 2005

Recently, embedded multimedia systems are often designed into SoCs based on IP cores. Most multimedia applications store large amount of picture data in the external memory. Therefore effective on-chip bus architecture is required to handle large memory access. In this paper, novel wrapper-based on-chip bus architecture is proposed for embedded multimedia SoCs. It has a separate memory bus so that it can handle large memory bandwidth and frequent memory access. It has a pipelined architecture to support multitasking on a single layer bus, i.e. in some cases several masters can access several slaves simultaneously. It has a probability-based arbitration policy to avoid starvation of low-priority masters.

2 Problems of Conventional Bus Architectures

AMBA bus [2] is one of the most popular on-chip bus architecture in IP-based embedded SoCs. It consists of AHB bus and APB bus, where the former is designed for high-performance operation and the latter is designed for low-performance external devices. AMBA bus shows good performance in on-chip IPs. However, it suffers from the following problems when applied to multimedia applications such as MPEG.

AMBA bus shows serious performance degradation in external memory with long or variable latency time. For example, MPEG system often requires external SDRAM memory to store large picture data. Therefore, SDRAM is connected with AHB bus due to huge memory access. However, SDRAM is accessed so frequently in the multimedia applications, and it has long latency time from the master's request. In AMBA bus, other IPs cannot use the bus when one IP already occupies it. Therefore, the IP accessing SDRAM occupies AHB bus during most of the operation time, and other IPs hardly share AHB bus. This seriously reduces the efficiency of AMBA bus. To overcome this problem, AMBA bus provides multilayer scheme to support multi-masters-to-single-slave connection. However, as the number of masters increases, bus area becomes large and wiring scheme becomes complicated. Furthermore, in the multilayer scheme, complex arbiter hardware is required because AMBA bus has only one arbiter that decides the priority of multi-masters.

Sonics bus [4] makes arbitration based on time schedule (TDMA). When SDRAM is connected to Sonics bus, SDRAM interface can directly require a master to read or write in order to prevent delaying due to long latency time of SDRAM. Sonics bus is easier to optimize than AMBA bus, and IP reuse is also easier because it supports OCP. However, very complicated SDRAM interface is required to increase bus efficiency, and the very detailed design information of IPs is also required.

SAMBA bus [7] is a synchronous bi-directional bus with multitasking. It shows better bus bandwidth over AMBA. It also reduces performance degradation due to long arbitration latency. However, it shows long latency time to read data from external memory, and it is not suitable for multimedia applications.

In segment bus [8], the response of slow slaves may cause severe performance degradation because it remains idle while the master is waiting for the response. Split transactions of segment bus are used to avoid such performance degradation. However, in split transactions, the bus access right of the master is released after the slave obtains the communication request. The slave has to initiate a new bus transaction to transfer

the response. Therefore, split transactions lead to an increase in arbitration complexity. Also, it can not assure the compatibility of the developed IPs with the conventional chip architectures.

3 Proposed Bus Architecture

3.1 Overview

In this paper, we propose a hierarchical bus structure for multimedia applications. Fig. 1 illustrates the proposed bus architecture. It exploits probability-based arbitration policy to avoid starvation of low-priority masters, which is described in the next section. It consists of two separate buses, i.e. system bus (SBUS) and memory bus (MBUS). SBUS is a bi-directional bus with pipelined architecture to support multitasking. MBUS is a multi-masters-to-single-slave single layer bus, and it has an efficient communication protocol with external SDRAM memory. External SDRAM memory and IPs with large memory access is connected to MBUS. When an IP directly accesses SDRAM via MBUS, other IPs can communicate with each other via SBUS, and it significantly reduces the performance degradation from memory access. SBUS and MBUS communicate with each other via memory bus bridge (MBridge).

The proposed bus architecture has three types of arbiters. Local arbiter (LArbiter) is attached to each master in SBUS. It performs arbitration based on TDMA. The priority is calculated from the pre-design analysis of each master. Central arbiter (CArbiter) performs channel control of SBUS. It grants approval of each master according to the channel status when local arbiter requests channel access. Control signals of LArbiters and CArbiter have short latency time, since they are separate from command transfer signals, address transfer signals, and data transfer signals. LArbiters and CArbiter are not integrated in order that additional masters can be easily inserted. MBUS arbiter (MArbiter) controls MBUS when a master in MBUS accesses external SDRAM memory. When a master in SBUS accesses SDRAM, it communicates with MBridge via SBUS, and then MBridge accesses SDRAM via MBUS. MBridge acts as a slave in SBUS and a master in MBUS.

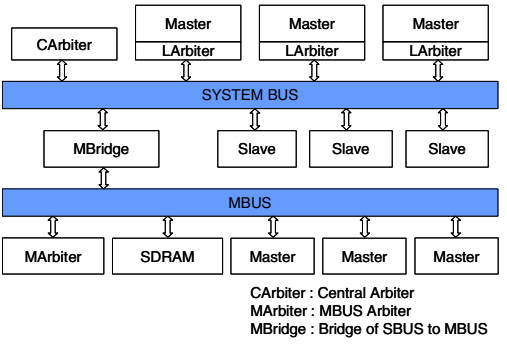
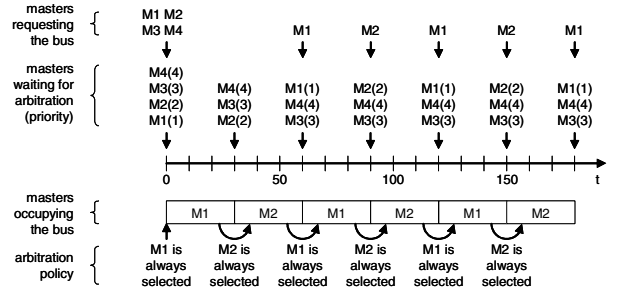


Fig. 1. Proposed bus architecture

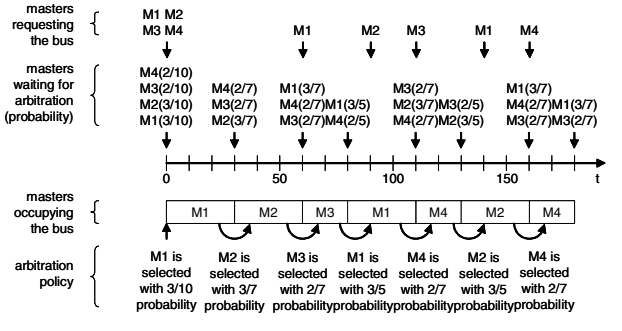
3.2 Probability-Based Arbitration Policy

AMBA bus exploits priority-based arbitration policy. Each master has its own fixed priority, and a master with higher priority is selected among waiting masters. Unfortunately, in AMBA bus, masters with lower priority sometimes fall into starvation. For example, assume that (1) there are four masters M_i (i=1..4) where lower i has higher priority, (2) their execution times are 30ms, 30ms, 20ms, and 20ms, respectively, and (3) they send next bus requests at 30ms after they finish current execution. As shown in Fig. 2 (a), when M_1, M_2, M_3, and M_4 send bus requests simultaneously at t=0ms, M_1 occupies the bus at t=0ms and M_2 occupies it at t=30ms. At t=60ms, M_1 sends next request and it occupies the bus again. At t=90ms, M_2 sends next request and it occupies the bus again. In this way, M_3 and M_4 can hardly access the bus and they fall into starvation, unless M_1 and M_2 stop sending their requests.

To avoid this problem, the proposed bus architecture exploits probability-based arbitration policy, where the arbitration probability of each master is determined in proportion to its execution time. In the previous example, M_1, M_2, M_3, and M_4 get the bus at t=0ms with the probability of 3/10, 3/10, 2/10, and 2/10, respectively, as illustrated in Fig. 2 (b). Assume that M_1 and M_2 are determined to get the bus at t=0ms and t=30ms, respectively. At t=60ms, M_1, M_3 and M_4 wait for bus arbitration, and M_3 and M_4 still have chance to get the bus with the probability of 2/7 and 2/7,



(a) Priority-based arbitration



(b) Probability-based arbitration

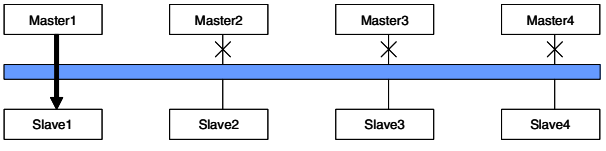
Fig. 2. Probability-based arbitration policy

respectively. Assume that M_3 and M_1 are determined to get the bus at t=60ms and t=80ms, respectively. At t=110ms, M_2, M_3 and M_4 wait for bus arbitration, and M_4 still have chance to get the bus with the probability of 2/7. Thus, starvation hardly occurs. The probability-based arbitration is implemented by simple logic gates with negligible hardware overhead.

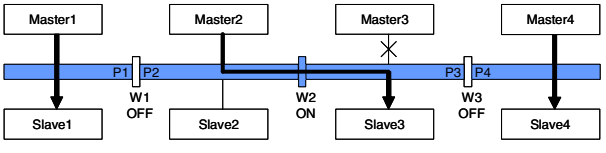
3.3 Bus Partitioning for Concurrent Multiple Master-to-Slave Communications

In AMBA bus, several masters and slaves are connected to a common bus. Consequently, when a master accesses a slave, other masters cannot access other slaves even if each master tries to access different slave, as shown in Fig. 3 (a).

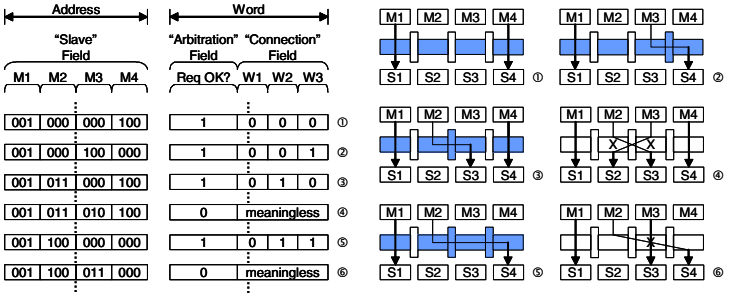
In the proposed bus architecture, we propose bus partitioning as illustrated in Fig. 3 (b). The proposed bus is partitioned into N partitions, where N is the number of masters. Each partition is connected via N-1 partition switches. When a switch is off, partitions in the left and right of the switch act as two separate buses. Therefore, more than two masters can access their slaves concurrently. In Fig. 3 (b), there are 4 masters (master1, master2, master3, and master4) and 4 slaves (slave1, slave2, slave3, and slave4). The bus is divided into 4 partitions (P1, P2, P3, and P4), and they are connected via 3 partition switches (W1, W2, and W3). When master1, master2, and



(a) Non-partitioned bus



(b) Bus partitioning



(c) ROM contents and bus operation

Fig. 3. Bus partitioning for concurrent multiple master-to-slave communications

master4 try to access slave1, slave3, and slave4, respectively, the bus arbiter opens W1 and W3, and it closes W2. Thus, the bus acts as three separate buses (P1, P2+P3, and P4), and three masters access three slaves concurrently (master1→slave1, master2→slave3, and master3→slave4).

Proposed bus arbitration is performed as follows. Assume the followings: (1) there are M masters denoting master$i$ (i=1..M) and S slaves denoting slave$i$ (i=1..S), (2) the bus is divided into max(M, S) partitions denoting Pi (i=1..max(M, S)) and they are connected via max(M, S) – 1 switches denoting Wi (i=1..max(M, S) – 1), (3) c connections denoting $C_i(m_i, s_i)$ (i=1..c, c<M) are made currently, where $C_i(m_i, s_i)$ indicates that masterm_i is connected to slaves_i. When masterm_k tries to access slavem_k, it can get bus arbitration when Eqn. (1) is satisfied. Otherwise, masterm_k should wait for bus arbitration until Eqn. (1) is satisfied.

($m_k > m_i$ and $s_k > s_i$) or ($m_k < m_i$ and $s_k < s_i$) for all connections C_i (i=1..c, c<M) (1)

When masterm_k gets bus arbitration, the bus arbiter opens W*left* and W*right*, and it closes W*mid*, where *left*, *right*, and *mid* are given as Eqn. (2)-(4).

$$left = \min(m_k, s_k) \tag{2}$$

$$right = \max(m_k, s_k) \tag{3}$$

$$mid = (left + 1) \,..\, (right - 1) \tag{4}$$

Practically, the proposed bus arbitration is implemented by simple ROM as illustrated in Fig. 3 (c). ROM address is $\{M \times \log_2(S+1)\}$ bits and ROM word is max(M, S) bits, where M and S are the number of masters and slaves, respectively. 'Slave' field indicates the slave ID in connection, and 0 means that the corresponding master accesses no slave currently. 'Arbitration' field indicates whether the requested access is possible or not. 0 means that such request is impossible and the requested master should wait for next chance. 'Switch' field indicates the partition switch status, and 0 means that the corresponding partition switch is off. When there are 4 masters and 4 slaves, ROM size is only 12bit address × 4 bit word, which is quite small. Even when there are 16 masters and 16 slaves, ROM size is only 80bit address × 16 bit word, which is quite tolerable.

3.4 Bus Architecture

Fig. 4 shows the SBUS architecture for command signals, address signals, and data signals. When a master transmits a request, LArbiter sends a 'Request' signal to CArbiter. After CArbiter checks channel status, it sends 'Done' and 'Accept' signals to the master. Simultaneously, channel gate signals of CArbiter control channel gates with pipelined architecture. Dashed block in Fig. 4 is a transmission control unit for command signals, address signals, and data signals. It consists of multiplexers, flip-flops, and tri-state buffers. It exploits bidirectional transmission for wiring efficiency. Pipelined structure with flip-flops is also employed for performance improvement and accurate data transmission. Masters in SBUS communicate with SDRAM via

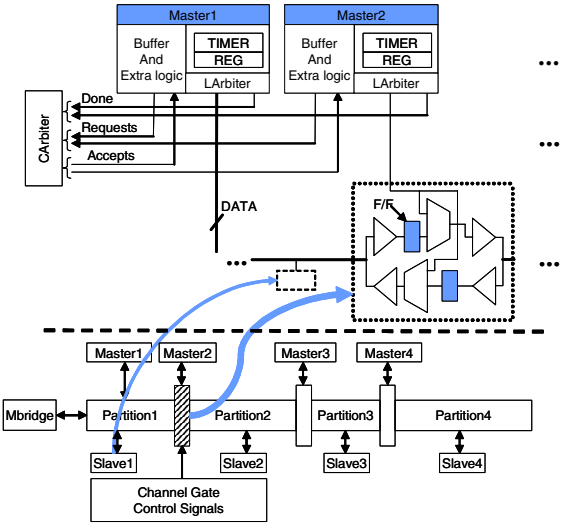


Fig. 4. SBUS architecture

MBridge and access latency time is different for each master. Leftmost master in SBUS has minimum latency time, so the master accessing SDRAM most frequently should be assigned to partition1. Therefore, when designing SBUS, masters should be assigned according to the descending order of SDRAM access frequency.

When LArbiter of a master sends 'request' signal to CArbiter, CArbiter looks up its internal ROM. When its 'arbitration' field is 1, CArbiter sends channel gate control signals to open or close partition switches. Partition switch is tri-state buffers in Fig. 4, and CArbiter directly controls it to reduce latency time. At the same time, it sends 'accept' signal to the master. Then, the master begins data transmission.

3.5 Bus Operation

There are two modes in SBUS bus protocols, i.e. request mode and response mode. Request mode is used when a master sends data to a slave, and response mode is used when a slave sends data to a master. These two modes have symmetrical structures with command bus, address bus, and data bus. SBUS has four direct control signals. DCS1 is 'request' signals from LArbiter to CArbiter. DCS2 is 'accept' signals from CArbiter to master. DCS3 is 'done' signals from slave to CArbiter. DCS4 is channel control gate signals (CGCS) from CArbiter to partition switch. Bit width of DCS1, DCS2, DCS3, and DCS4 are M bits, M bits, S bits, and $4P$ bits, respectively, where M, S, P are the number of masters, slaves, and partitions, respectively.

Fig. 5 illustrates data flow of concurrent multiple master-to-slave communications when master1 sends data to slave2 and master3 sends data to slave4. When master1 tries to send data to slave3, master1 sends 'request' signal to CArbiter (①). CArbiter checks if such arbitration is possible, and sends 'accept' signals to master1 (②). At the same time, CArbiter sends CGCS to open or close partition switches (②). In this case, partition gates W1, W2, and W3 are on, off, and off, respectively. After that,

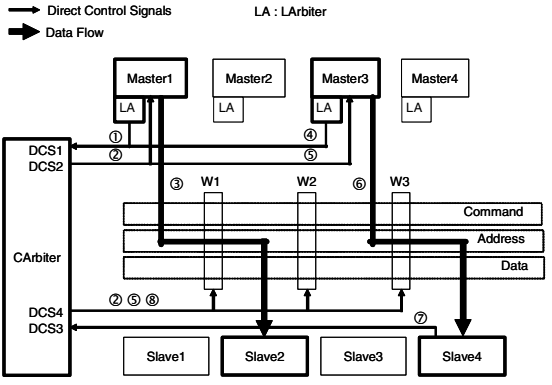


Fig. 5. Data flow of concurrent multiple master-to-slave communications

data is transmitted from master1 to slave2 (③).When master3 tries to send data to slave4, master3 sends 'request' signal to CArbiter (④). CArbiter checks if such arbitration is possible, and sends 'accept' signals to master3 (⑤). At the same time, CArbiter updates CGCS to open or close partition switches (⑤). In this case, partition gates W1, W2, and W3 are on, off, and on, respectively. After that, data is transmitted from master3 to slave4 (⑥). When the transmission from master3 to slave4 finishes, slave4 sends 'done' signal to CArbiter (⑦). Then CArbiter updates CGCS to open or close partition switches (⑧). In this case, partition gates W1, W2, and W3 are on, off, and off, respectively.

4 Simulation Results

To evaluate the performance of the proposed bus architecture, we designed both AMBA bus and the proposed bus, and implemented them with Verilog HDL. They were verified and simulated on hardware level, and their performances were evaluated for comparison. We used SDRAM as external memory.

The number of masters, slaves, and partition are assumed as 3, 4, and 4, respectively. We assumed that the workload of master1, master2, master3, and master4 are 40%, 30%, 20%, and 10%, respectively. We used three test patterns, i.e. uniform traffic pattern, non-uniform traffic pattern, and random traffic pattern. In the uniform traffic pattern, each master accesses 4 slaves equally. Master1 equally accesses slave1, slave2, slave3, and slave4 in 25%. Master2, master3, and master4 access slaves in a same way. In the non-uniform traffic pattern, each master accesses adjacent slave more frequently than other slaves. Master1 accesses slave1 in 85% and it equallyaccesses slave2, slave3, and slave4 in 5%. Master2 accesses slave2 in 85% and it equally accesses slave1, slave3, and slave4 in 5%. Master3 and master4 access slaves in a same way. In the random traffic pattern, each master accesses 4 slave randomly. Bit width of data bus and address bus is 32 bits. TDMA is exploited as priority scheme. In multimedia applications, large amount of picture data is sequentially accessed in frame memory, so the transmission is modeled as burst mode

with 16 increments. Inter-communication interval is 3, 6, and 9. We used two performance measures to evaluate the proposed bus architecture. Normalized effective bandwidth is defined as the normalized value of (amount of transmitted data) / (elapsed bus cycles) with AMBA bus. Average communication latency is the average of latency cycles from bus request to bus accept.

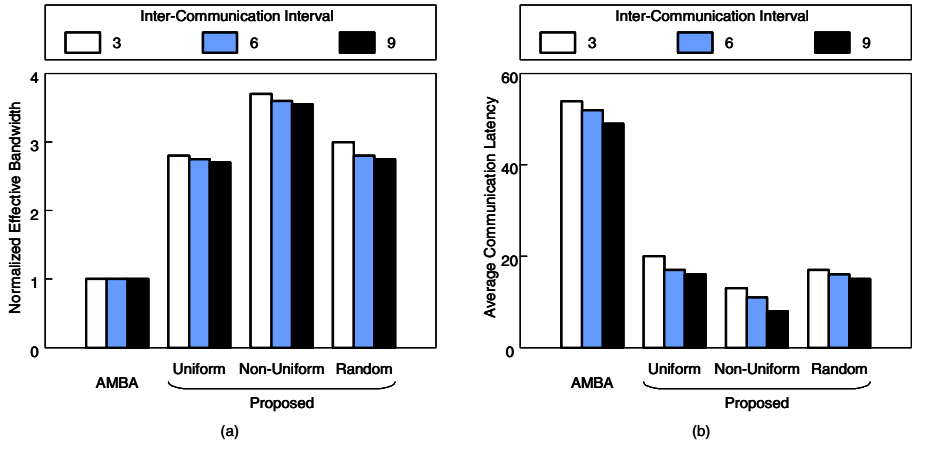


Fig. 6. Simulation results (a) Normalized effective bandwidth (b) Average communication latency

Fig. 6 shows the simulation results. Both AMBA bus and the proposed bus are designed in Verilog HDL and synthesized on Synopsys CAD tools supported by IC Design Education Center. The bus operation is simulated in Synopsys CAD tools, not in C++. Therefore, the simulation results consider all hardware effects including gate delay, bus line delay, buffer operation, and pipelining. Note that the simulation results of AMBA bus are identical regardless of traffic patterns. In AMBA bus, only one master can access a slave at a time and there is no difference in bus performance which slave a master accesses. We performed 320 bursts from every 4 masters to every 4 slaves, and the amount of transmitted data is 320 (burst number) × 32 bits (bandwidth) × 16 (16 increments burst) × 4 (masters) × 4 (slaves) = 327,680 bytes. In this case, AMBA bus takes about 20,000 cycles. On the contrary, the proposed bus takes about 7,250 cycles in the uniform traffic pattern. Consequently, the proposed bus achieves 2.8 times speed-up in the uniform traffic pattern. Similarly, it achieves 3.6 and 2.8 times speed-up in the non-uniform traffic pattern and random traffic pattern, respectively. As for average communication latency, the proposed bus achieves 3.1, 4.7, and 3.3 times speed-up in the uniform traffic pattern, non-uniform traffic pattern, and random traffic pattern, respectively. In the simulation results, the proposed bus shows best performance improvement in the non-uniform traffic pattern. This is due to the fact that the bus partitioning shows better performance when a master accesses its adjacent slaves more frequently than other slaves.

5 Conclusion

This paper proposes novel bus architecture for embedded multimedia SoCs. It consists of two separate buses called system bus (SBUS) and memory bus (MBUS) for effective memory access. It effectively handles huge memory bandwidth and frequent memory access of multimedia applications, since most memory-intensive IPs are connected to MBUS and they hardly disturb SBUS operations. The proposed bus architecture exploits probability-based arbitration policy, and masters with low priority hardly fall into starvation. It exploits bus partitioning to enable concurrent multiple master-to-slave communications to increase transmission bandwidth. It exploits pipelined bidirectional architecture to achieve high performance. It supports VCI and OCP to reuse IPs easily. The proposed bus architecture is designed, implemented, verified, and evaluated in hardware level. From the simulation results, it shows 2.8~3.6 times speed-up in effective bandwidth and 3.1~4.7 times speed-up in communication latency. Especially, it shows significant performance improvement when each master accesses its adjacent slave than other slave, which is typical in embedded multimedia SoCs.

References

1. Keating, M., Bricaud, P.: Reuse Methodology Manual for System-on-a-Chip Designs, Kluwer Academic Publishers (1998).
2. ARM: AMBA Specification Overview, http://www.arm.com/ Pro+Peripherals/AMBA.
3. IBM: CoreConnect Bus Architecture, http://www.chips.ibm.com/products/coreconnect/ docs/ cron_wp.pdf.
4. Sonics: Sonics μNetwork Technical Overview, http: http//:www.sonicsinc.com/Documents/ Overview.pdf.
5. OCP International Partnership: Open Core Protocol Specification, http://www.ocpip.org.
6. Anjo, K., Okamura, A., Motomura, M.: Wrapper-Based Bus Implementation Techniques for Performance Improvement and Cost Reduction, IEEE Journal of Solid-State Circuits **35** (2004) 804-817.
7. Lu, R., Koh, C.-K.: SAMBA-bus: A High Performance Bus Architecture for System-on-Chips, Proceedings of International Conference on Computer-Aided Design (2003) 8-12.
8. Plosila, J., Seceleanu, T., Liljeberg, P.: Implementation of a Self-Timed Segmented Bus, IEEE Design and Test of Computers **20** (2003) 44-45.

On Tools for Modeling High-Performance Embedded Systems

Anilkumar Nambiar and Vipin Chaudhary

Department of Computer Science, Wayne State University
{anambiar, vipin}@wayne.edu

Abstract. Most of the new embedded systems require high performance processors at low power. To cater to these needs, most semiconductor companies are designing multi-core processors, also known as chip-multiprocessors, while some are developing multi-chip boards with existing multi-core processors. Developing applications on these powerful architectures require specialized tools to obtain the optimum performance. Most applications running on these processors not only require high processing power, but also have tight resource constraints. For both, chip-multiprocessors and multi-chip boards, one faces some common problems while developing applications for them. To meet this end, we have developed tools to model high-performance embedded applications on these complex high-end systems. We have integrated these tools in the modeling framework of MILAN, and have modeled a real application, Mpeg-2 Audio Video Decoder. For validation we have used Cradle Technologies MDSP multi-core chip as the target processor.

1 Introduction

Chip Multiprocessors are being widely used to develop System-on-Chip (SOC) solutions for high-end embedded systems. An SOC consists of processors (RISC/DSP), internal memory, external memory, device controllers, timers, etc. SOC architecture for a low/medium scale device would not require more than a couple of processors. On the other hand, high-end embedded systems, such as HDTV, biomedical applications, high capacity printers and copiers would require multiple cores of RISC/DSP or a combination of both, to fulfill the high performance requirement. Most of these applications can be parallelized over multiple low power processors to obtain the desired performance.

Most of the multi-core chips available consist of not more than 8 cores on a single chip [1]. For a high-end application this might not be suitable. ASIC-based solutions are not suitable either, as these solutions take longer time to market and have high cost per unit. There are few companies developing multi-core chips with 30-60 cores in one chip [2] [3]. While other companies, in order to achieve the high performance requirements for these high-end applications, develop multi-chip boards, with each board housing 2-4 chips, where each chip has 4-8 processors [4][5][6]. This gives enough computational power for the high-end systems.

As the number of processors increase, the bandwidth requirement for the application over the shared global bus increases. In order to sustain such high

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 360 – 370, 2005.
© IFIP International Federation for Information Processing 2005

requirements, these systems have a hierarchical architecture consisting of clusters. Each cluster is made up of RISC and/or DSP processors with some shared internal memory to be used as a scratch pad. In addition, it has semaphores, timers and interrupts. Any access to the resource within the cluster takes less time as compared to any access to resources in the other cluster or global resources since the request has to be served through shared global bus. The desirable implementation on this system would be to reduce access outside the cluster in order to improve performance. The programmer can also benefit from the nature of most of the high-end applications, where the computation to communication ratio is high.

Due to similarity in architecture, these high processor-count chip-multiprocessors and multi-chip board solutions face some common problems for developing applications. Though these systems provide high computational power, they require specialized tools to harness their raw power. One requires tools to design and map resource constraint applications on these high performance systems, tools to simulate the system at various granularity levels and provide feedback through profiling statistics, which can be incorporated back into the design to optimize the application. As most of the high-end system would be heterogeneous, the tools must provide mechanisms or interfaces for seamless integration of different simulators for different types of processors in the underlying system.

There are lots of tools for modeling SOC systems. Some of them are Ptolemy[7], Polis [8], Chinook [9], SystemC [10] and MILAN [11]. We tried to use these tools for modeling systems on such clustered/hierarchical architecture and found that they were not suitable in their existing setup. Modification needs to be done to incorporate the type of architecture we were using. Most of these tools are complex and proved cumbersome to integrate the simulators for this architecture. MILAN is relatively easy to understand and provides simple interface to integrate any new simulator. It also provides Design Space Exploration capability to obtain a near optimum design solution for the system under consideration.

For validation, we have considered Cradle Technologies' MDSP architecture [12]. This architecture uses a combination of low performance RISC and DSP processors, with each chip consisting of 36 to 60 processors. This architecture provides a very high raw computational power in the range of 10-15 GMACs. We have developed an interpreter for MILAN to model applications for MDSP. In addition, we have developed and incorporated a new heuristic for mapping applications on clustered architecture. In this paper, we have presented modeling of a complex application Mpeg-2 Audio Video Decoder (MAVD) using MILAN on the MDSP chip.

The paper is organized as follows. Section 2 gives the problem definition, section 3 discusses the Model Interpreter, section 4 discusses the MAVD modeling, section 5 discusses the related work in this area and section 6 has the concluding remarks.

2 Problem Definition

The MDSP is a heterogeneous multi-core chip developed by Cradle Technologies. The chip has a hierarchical architecture, with clusters within a chip. Each cluster or Quad consists of 4 RISC processors or PEs, 8 DSP co-processors or DSEs, 64 Kbytes of data memory, 4 channel DMA engines and other resources such as semaphores,

and timers. Each chip may have varying number of clusters from 1 to 5. If one wants to design an MPEG-2 Audio Video Decoder (MAVD) system using the MDSP chip, one needs to decide on the resource requirements for each task. A typical MAVD system will consists of following tasks: Audio Decoder, Video Decoder, Audio Renderer, Video Renderer, TS Parser, System Controller, TS Feeder Simulator. The memory and processor requirement provided by Cradle Technologies is given in the Fig. 1.

Program and its Resources	Audio Deco-der	Audio Rend-erer	Video Deco-der	Video Rend-erer	TS Parser	System Cont-roller	TS Feeder Simulato	ISR Simu-lator	Timer Simu-lator
Compu-tation	1 PE 1 DSE (1 data DSE)	1 PE 1 DSE	6 PE 12 DSE	1 PE 2 DSE	1 PE 1 DSE	1 PE	1 PE	1 DSE	1 DSE
Private Local Memory	0x22D0	0xDD0	0xD68	0xBF8	0xED8	0x2F0	0x78	-	-
Shared Local Memory	-	-	0x9C0	-	0x8	-	-	-	-

Fig. 1. MAVD System Resource Requirement Table

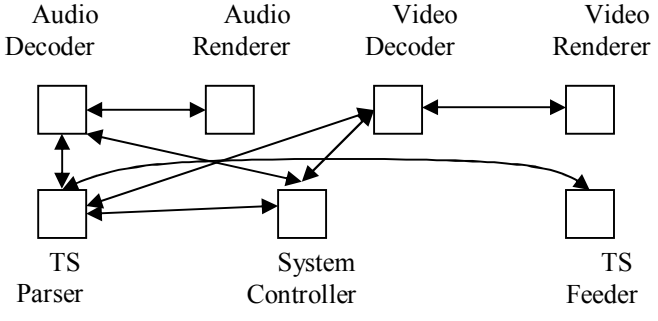


Fig. 2. MAVD System Communication Graph

While designing such a complex system, the extent of communication between different modules might not be obvious. One would like to select a design, which would minimize the execution time. Most of the time the system tasks are developed independently and needs to be integrated. During integration, one would like to map the system on the chip, which would reduce the communication time, which in turn would minimize the execution time. MILAN provides excellent design space exploration functionality, whereby one can consider different designs, simulate those at high level or low level using the appropriate architecture simulators and then choose a good solution. In our case, since we are only considering one chip, the simulation becomes simple.

MILAN incorporates an optimal mapping algorithm for linear array of tasks [13]. But it requires the tasks to be executed in linear order, whereas in our case, the tasks are executed concurrently on different processors. Hence it cannot be used for our problem domain. In order to support MDSP chip in MILAN, one needs to provide:

a. MILAN interpreter to interpret system models and generate output for MDSP simulator, and associated tools.
b. A new mapping heuristic to map applications on MDSP processor.

3 Model Interpreter

MILAN uses GME or Generic Modeling Environment [14] to synthesize a domain specific modeling interface for modeling the system. GME provides interfaces to specify different modeling paradigm for different domains. Modeling paradigm is a set of building blocks and composition rules using which, the designer models the system. MILAN is an example of such a modeling paradigm.

While modeling any system, the designer will perform two main activities:

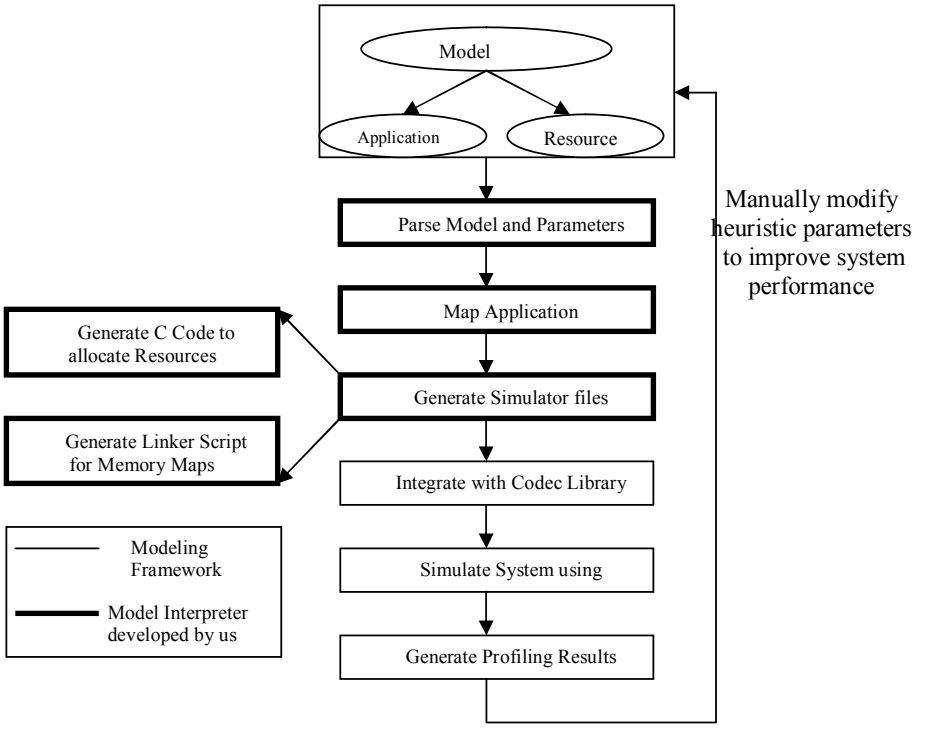


Fig. 3. System Modeling

1. Application Modeling, where the designer defines application modules or tasks and inter-task communication between different modules using task-graph.
2. Resource modeling, where the designer models the underlying architecture. In this the designer defines various architecture parameters such as frequency, power consumption, cache size etc. for the architecture under consideration.

In application modeling, we found no provision to specify shared memory communication between multiple modules. In the existing setup in order to represent shared memory communication one would have to show it by connecting different modules using communication channels. This might complicate the design with lots of connections between modules involved in communication. In order to represent shared memory communication, we added a new feature to the paradigm to represent shared memory communication.

MILAN provides various model interpreters to interpret the model and generate output for different simulators. In order to generate the output for MDSP simulator, we developed our own interpreter. This interpreter takes the model and applies our proposed heuristic to generate a good mapping scheme and then generate code to allocate resources and launch the specific threads of various tasks on the processors. This divides the interpreter into three main components, the Model Parser, the Mapping Heuristic and the Code Generator.

3.1 Model Parser

The Model parser is a COM object, which is invoked from the GME as an interpreter. This parser performs a top-down parsing of the components defined in the GME and forms the input to the Mapping heuristics. It forms the list of tasks T defined, based on primitive components. It also forms the resource requirement matrix and communication matrix. The priority for each task is obtained from the attributes of the task. The processor requirement for each task is obtained from the resource reference provided by the user in the Mapping Aspect window. These inputs are provided to the Mapping heuristic module to provide a good mapping scheme.

3.2 Mapping Heuristic

MILAN provides a mapping algorithm to map linear array of tasks onto single and multiple device. It requires the tasks to be in linear order where, given a set of tasks T = $\{t_1, t_2, \ldots t_n\}$, t_{i+1} executes only after t_i, $1 \leq i \leq n$. The mapping problem in our case is different than the one discussed.

Suppose we want to map the MAVD system on the MDSP processor by integrating different tasks, which were developed independently. In this case some of the tasks are independent, while others have inter-task communication. The first requirement would be to allocate all the processors for a given task in the same cluster, as most of the tasks will have more intra-task communication than inter-task communication. In addition to intra-task communication, the tasks will communicate with the allocated resources to varying degree. The resources in this case are memory, and co-processor. If there were more communication between co-processor and processor than inter-task communication, it would be better to allocate the co-processor from the same

cluster in which the task is running, and then try to see if the two tasks involved in communication can be allocated to the same cluster.

We have developed Improved First Fit Decreasing [15], which improves over the existing First Fit Decreasing (FFD) algorithm [16]. Since we use a priority queue, if there are *m* clusters then this will require *O(nlogn*m)* time, where *n* is the number of tasks to be mapped. So, in worst case it would be $O(n^2 logn)$, where the number of tasks is equal to the number of clusters in the chip.

3.3 Code Generation Module

The code generation module generates C code to allocate processors and launch programs on the processors. To compile a multi-task program on MDSP, one needs a loader script also known as *LD* script to determine the code sections and the memory requirement for the executable. This *LD* script is used by the GNU *ld* program to generate the executable. The script specifies the tasks running with information about the local and DRAM data memory used by each task. In addition it also specifies the code sections for each task.

For any program running on MDSP, the main thread is started on QUAD 0, PE 0. This thread will start the other threads on the other processors. The task, which is mapped to QUAD 0, PE 0 forms the main thread. This thread will launch one thread from each of the other tasks on the mapped processors. Those threads in turn will start threads from its task. The launching is similar to a tree structure and its shown in the following figure.

The code in table 1 is from the main task launched on QUAD 0 PE 0.

During application modeling one specifies the buffer length to be used in the communication between different modules. This buffer length is used to allocate the shared memory structure. A shared memory structure consists of the following fields, Memory ID, Semaphore, Thread Count, Buffer Size, and Buffer Pointer.

Based on the tasks involved in communication the buffer will be allocated in local memory or shared DRAM. If all the tasks involved in communication are in the same cluster, then the shared memory is allocated in the local memory of the cluster. This would reduce the communication latency as the local memory access takes far less amount of time than DRAM. But, this is based on the availability of the local memory.

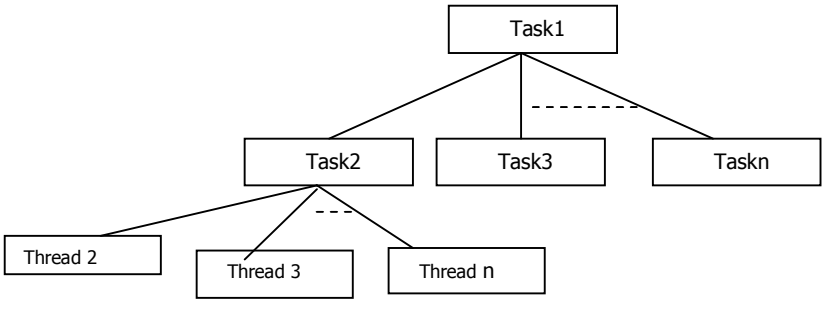


Fig. 4. Task Startup Tree

If the tasks involved in communication are in different cluster, then the buffer is allocated in the Shared DRAM. Some communication buffers require to be guarded by a semaphore. Again, based on the tasks involved in communication, the semaphores are allocated from the local or global pool. The thread count indicates the number of threads involved in communication. The buffer can be freed when the thread count goes to zero.

For shared memory communication one has to generate structures to be used with the shared memory library. We developed this library, and it has a similar structure to that of the windows shared memory library API. The shared memory library structures are generated in common header files. Once this code is generated, the user can integrate it with the core Intellectual Property modules developed. In the case of MAVD, the audio and video library forms the IP libraries.

This code is compiled and linked using the LD script generated to produce the executable. This executable is run on the hardware or the simulator to verify the results.

Table 1. Code generated by the Mapping tool for one of the synthetic test application

```
#define MAX_PROCS 20          /*total processors on the chip*/
void main()
{
        int procArray[MAX_PROCS] = {0,0,0,5,6,6,6,6,2,2,2,5,3,3,3,7,1,1,4,4};  /* application number
        loaded on proc */
        int my_peid;
        my_peid = (_QUAD_INDEX * 4) + _PE_INDEX;
        if(my_peid == 0)
        {
                StartPrograms(procArray);          /* start 1  instance of each task*/
                start_other_pes(0, 2, procArray); /* other instances of this task */
        }       my_peid = get_PE_number();  /* get my PE ID*/
        /*Perform useful computation*/
        terminate_program(my_peid, (Load_Table *)load_table_address, FREE_PL | FREE_PD); /*
        terminate the task*/
}
StartPrograms(int *procArray)
{
        int i;
        for(i = 1; i < MAX_PROCS; i++) /*  for all the tasks*/
        {
                if(procArray[i] != -1)
                {
                        if(procArray[i-1] != procArray[i]) /* start only one instance of this task*/
                                start_program_on_proc(procArray[i], i, 0);
                }
        }
}
```

4 MAVD Modeling

The two main types of modeling in MILAN are Application Modeling and Resource Modeling. During application modeling, one builds the dataflow model in the dataflow aspect window. There are three main types of dataflow classes; Primitive,

Compound, and Alternate. Compounds are composite dataflow nodes. They contain dataflow graphs themselves. For example, in MAVD, the Audio Decoder is a composite model. It will contain the PE component and the DSE component, which forms the primitive model. Alternates are similar to composite models, but they represent alternative implementation. Suppose a component can be executed on PE and on DSE then one would specify it as an alternate. During mapping one can select the processor based on the availability. For system instantiation only one alternate is selected. Figure 5 shows the MAVD model created in GME using MILAN paradigm. On the right hand window the folder structure is shown. MAVD is the root composite folder with different modules in it. These folders in the root folder are composite module and form root for the module. The communication between different modules are bi-directional and is shown using connection between in-port and out-port of the specific modules.

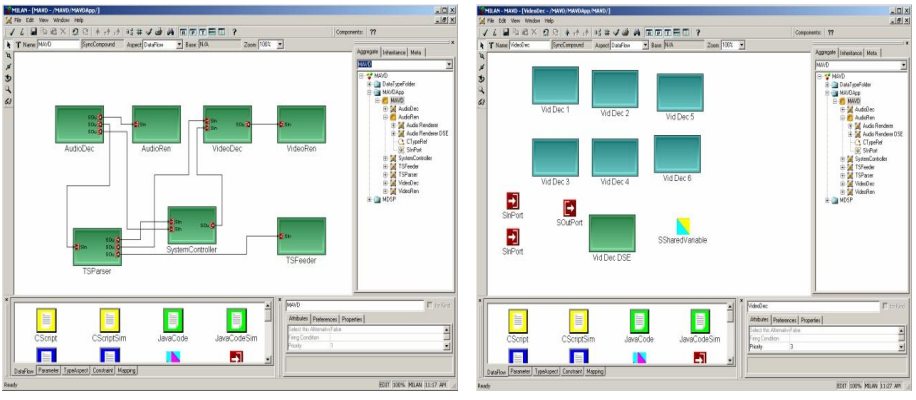


Fig. 5. Modeling MAVD using GME

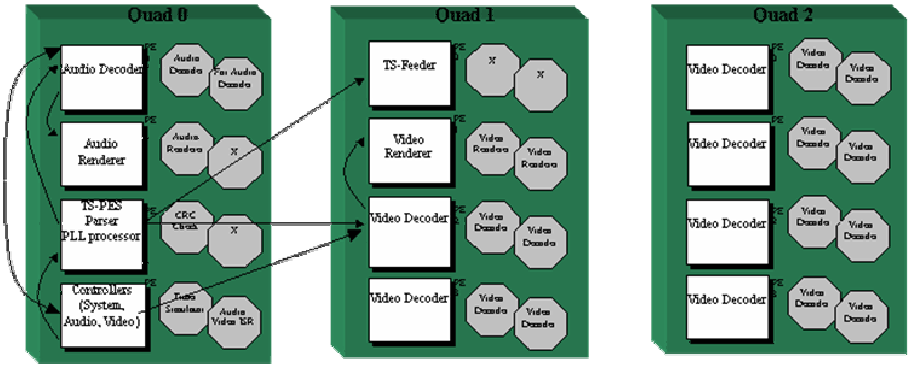


Fig. 6. MAVD Solution for MDSP Chip

For each of the composite module there are primitive or composite modules within these modules. Figure 6 shows the Video Decoder Module. There are 6 Video Decoder threads running on 6 PEs and 12 Video Decoder threads running on 12 DSEs. The 6 PE threads are shown as 6 primitive components, while the 12 DSE threads are enclosed within a composite component, which has 12 primitive components. This makes design easy to understand.

In dataflow model there are two ways of showing shared memory communication. One way is to show it through connection between all the modules. This becomes complicated with too many connections. The other way is add a new primitive called SharedMemory in the window, which will show the shared memory communication between all the modules shown in the current window. We have defined a new Atom [13] called SSharedVariable in the MILAN paradigm.

Once a primitive component is defined, it needs to be mapped onto one type of processor. But before we map onto processor, we need to model these processors. This is done using the Resource Modeling paradigm in MILAN. MILAN provides support for resource modeling to model various architecture capabilities that can be exploited to perform design space exploration and drive a set of widely used energy and latency simulators from a single model. The Resource Modeling provides support for modeling of various architectural components such as CPU, BUS, Memory, IO. One can even specify the type of cache used in the architecture. For each component one can specify what is the throughput, energy estimate, latency estimate and various other parameters. To specify that the current primitive module is going to run on a particular processor, one has to provide a reference to the processor in the primitive module in the mapping aspect.

Figure 6 shows the final mappin. In the MAVD system, the Video Decoder and Video Renderer has more communication, so they are placed within the same cluster. Similarly, Audio Decoder and Audio Renderer are placed in the same cluster. There is more communication between TS-PES Parser task and Audio Decoder task, so they have to be placed in the same cluster. Similarly, the System Controller has more communication with Audio Decoder due to higher rater of Audio ISR than Video ISR and needs to be placed in the same cluster.

We compared the mapping obtained by our algorithm and the mapping done manually by experienced system designers from Cradle over the span of two years and found it to be same. The design had gone through multiple iterations before it was finalized, while we obtained it using our tool within the first iteration.

5 Related Work

One of the most popular hardware/software co-design tools is Ptolemy. It provides an environment for design, modeling, and simulation of concurrent heterogeneous embedded systems. Though Ptolemy provides different models of computation, the main problem with it is, it does not provide mechanisms for integration of external simulators.

Writing simulators of complex multiprocessor systems for the modeling environment is not a trivial task. Instead, one can use the existing simulators provided with these processors and use them in the modeling environment. This is precisely

what MILAN offers. Easy integration of existing simulators, ease in modification of the paradigm to suit the underlying architecture.

There are other co-design or modeling projects for embedded system. Some of them are Polis [8], which uses finite state machine model for hardware/software co-design, Chinook [8], which focuses on IP composition, communication synthesis and rapid evaluation. In Chinook, one has to provide behavioral description of the application, target description and mapping function in order to synthesize the system. But none of these tools could suffice our need of simulator integration. The mechanism for mapping application onto the architecture was not present in Chinook. Instead, the user has to provide the allocation function. Though Polis provides mapping mechanism, it was not for clustered architecture like the one we are considering.

MILAN incorporates an optimal mapping algorithm for linear array of tasks [13]. But it requires the tasks to be executed in linear order, whereas in our case, the tasks are executed concurrently on different processors. Hence it cannot be used for our problem domain. There has been lot of research in scheduling theory for various types of problems [17] [18][19][20]. The solutions from these researches could not be directly applied to the problem under consideration.

6 Conclusions

In this paper we have presented tools to model complex high-performance embedded systems. We have validated our techniques by modeling a complex application using a multi-core processor. The same techniques can be used to model complex applications on multi-chip boards.

The tools still require man-in-loop process to provide feedback to the modeling environment after simulation, based on the profiling results. The future direction of the research would be to automate this process, so that given a system, the tools would optimize the design automatically without much human intervention.

References

1. Arm Corporation, www.arm.com
2. Cradle Technologies, www.cradle.com
3. Azul Systems, www.azulsystems.com
4. Kestral Multiprocessor, http://www.soe.ucsc.edu/research/kestrel/
5. Orion Multisystems, www.orionmulti.com
6. ClearSpeed, www.clearspeed.com
7. Davis J. et al. , Overview of the Ptolemy project, ERL Technical Report CB/ERL No. M99/37, Dept EECS, University of California Berkeley, July 1999
8. Balarin F. et. al. : Hardware-Software Co-Design of embedded Systems: The POLIS Approach, *Kluwer Academic Publisher* 1997
9. Chou P. et. al.: IPChinook: An integrated IP-based design framework for Distributed Embedded Systems, *DAC* 1999
10. SystemC user Guide, http://www.systemc.org

11. “MILAN: A Model Based Integrated Simulation Framework for Design of Embedded Systems”, Agrawal A., Bakshi A., Davis J., Eames B., Ledeczi A., Mohanty S., Mathur V., Neema S., Nordstrom G., Prasanna V., Raghavendra, C., Singh M. *Workshop on Languages, Compilers, and Tools for Embedded Systems (LCTES 2001),* Snowbird, Utah, June 2001
12. “3400 Hardware Architecture Reference”, Cradle Technologies, www.cradle.com.
13. Milan Manual, http://milan.usc.edu
14. Ledeczi A., et.al., The Generic Modeling Environment, *Workshop on Intelligent Signal Processing* , Budapest, Hungary, May 17, 2001
15. A. Nambiar and V. Chaudhary, “Mapping Resource Constrained Applications on Chip Multiprocessors”, *Proceedings of the 2005 International Conference on Embedded Systems & Applications (ESA'05)*, Las Vegas, Nevada, June 2005.
16. E. G. Coffman, M. R. Garey, D. S. Johnson “Approximation algorithm for Bin Packing: A Survey, Appears in Approximation Algorithms for NP-Hard Problems”, D. Hochbaum(ed) PWS Publishing(1996)
17. V. Chaudhary and J. K. Aggarwal, ``A generalized scheme for mapping parallel algorithms", in *IEEE Trans. on Parallel and Distributed Systems*, Mar '93, pp. 328 - 346.
18. D. G. Feitelson and L. Rudolph, “Distributed hierarchical control for parallel processing.” *Computer 23(5), pp. 65-77 , May 1990*
19. Dror G. Feitelson, “Packing Schemes for Gang Scheduling”, *Job Scheduling Strategies for Parallel Processing -- Proceedings of the IPPS'96 Workshop* (1996)
20. Y.-K. Kwok, I. Ahmad. “Dynamic critical-path scheduling: An effective technique for allocating task graphs to multiprocessors.” *IEEE Trans. on Parallel and Distributed Systems, vol.7, pages 506--521,* March 1996

A Hardware/Software Co-design and Co-verification on a Novel Embedded Object-Oriented Processor

Chi Hang Yau, Yi Yu Tan, Pak Lun Mok, Wing Shing Yu, and Anthony S. Fong

Department of Electronic Engineering, City University of Hong Kong,
Tat Chee Avenue, Hong Kong
Anthony.Fong@cityu.edu.hk

Abstract. In the past, programming language are procedural, the design concept is based on the module and scope which are difficult to manage, but nowadays, the programming trend is Object-Oriented Programming (OOP), where objects are the key elements to build up application and the communications between different objects are through method invocation. A novel object-oriented processor offers an opportunity to enhance the system security, performance and provides a more effective way to manipulate OOP instead of using a software Virtual Machine. jHISC is a novel object-oriented processor which provides a natural way to map the concept of OOP into architectural level through the hardware object data structure. Our solution is to design secure hardware object data structures on a novel processor with Just-In-Time compilation for Java which then makes it possible to implement complex OO related bytecodes at hardware level and access some fields of object in parallel to improve the execution speed. It mainly targets J2ME and implements about 93% bytecodes and 83% OO related bytecodes in hardware directly.

1 Introduction

Computer hardware becomes quite mature, however, software trend is moving to Object-Oriented Programming (OOP) with advantages on reusable design and better security. In traditional systems, processor architecture like Complex Instruction Set Computer (CISC) or Reduced Instruction Set Computer (RISC) does not support object manipulation directly on hardware. To support object technology in nowadays system, there are mainly three categories, which are compiling objects into native (Fig. 1), developing software-based object virtual machine (Fig. 1) and developing an object-oriented processor (Fig. 2).

For the first approach, since traditional computer system does not support object-oriented programming, applications written in object-oriented programming languages like C++ are compiled into native instructions for execution. Application processes will be created and executed in their own spaces, and they are invisible from each other through the use of virtual memory system. Through the specific compiler, objects will be translated into subroutines, which are machine readable for direct execution. Compilation approach can fully optimize the generated codes, but object-oriented features, such as dynamicity behavior, will be removed. Modification of a single class requires the whole application to be re-compiled.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 371 – 380, 2005.
© IFIP International Federation for Information Processing 2005

In the second approach, an object virtual machine application is built on top of the traditional operating system. From the view of operating system, the application is just a usual process like other applications. Protection of different processes is the same as in compilation approach and objects are manipulated on this software virtual machine. The advantages of this method are that without the modification of hardware platform and current system, object technology can be supported through software emulation. But the two layers of software, virtual machine and operating system, also introduce much overhead to the overall computing system and increase the memory footprint of the system.

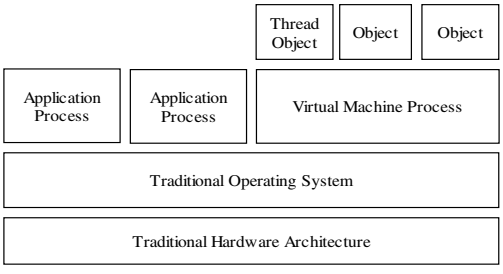


Fig. 1. System Architecture of Compilation Approach and Virtual Machine Approach

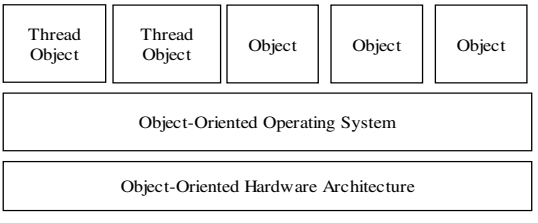


Fig. 2. System Architecture of Object-Oriented Hardware Approach

In our solution, we will use the third approach where the objects are directly manipulated on an object-oriented processor, and an object-oriented operating system will be built to assist hardware in managing the heap (Fig 2). In this approach, protection of objects is governed by the object-oriented operating system with the protection features offered by the object-oriented processor. This approach can manipulate objects directly and the efficiency of the computing system is increased because no page table updating is required during context switching.

In the following sections, we will firstly discuss our secure hardware object architecture on section 2, their overall security features on section 3. Moreover, we will show the hardware/software co-verification methodology which verifies the Java compatibility on our novel architecture, through the Just-In-time compilation on Java bytecode and demonstrate some co-verification results on section 4. Finally, a conclusion will be presented on Section 5.

2 jHISC v3 Architecture with Object Representation

In Java, objects are created with some class templates, which provide information of the object instance, such as methods, fields, etc. Object instances of the same class share the same method codes, but each instance maintains its own copy of instance variables.

There are different kinds of objects in an object-oriented system. An object can have three contexts which are the instance context, the class context, and the method context. In jHISC v3, it maps the three contexts to architecture. The information about objects in different contexts is stored in different data structures which are defined by the architecture for object representation. Generally, all the objects share the same extendable object data structure. Array and thread objects, which require additional support by hardware, are represented differently.

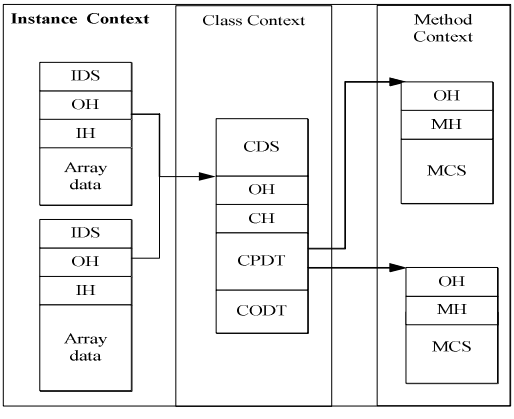


Fig. 3. Object Data Structure relationship

Three headers are defined. They are object, class, and method headers. These headers store information for describing the object, class, and method itself. A header may associate with other data structures for describing different aspects of the context.

Descriptor tables store descriptors. Each descriptor contains information about an element within the class. Two kinds of descriptor table are defined. They are Class Operand Descriptor Table (CODT), and Class Property Descriptor Table (CPDT).

Data spaces provide spaces for the storage of data. There are three types of data space. They are the Class Data Space (CDS), the Instance Data Space (IDS) and the Local Variable Frame (LVF). Unlike CDS and IDS, LVF are dynamically created inside the register file upon method invocation and destroyed upon method revocation.

Inside the method context, Method Code Space (MCS) is defined to store the instructions. Fig 3 shows the relationships between different data structures. On the following of this section, we will show each of our hardware data structure in details.

2.1 Descriptor Format

jHISC uses a 32-bit operand descriptors to describe properties own by a class or resources accessed by the class. Descriptor is a data structure which stores the information about a variable, reference or any data structures. In jHISC v3, a single descriptor format is defined, which combined both fields in Class Operand Descriptor (COD) and Class Property Descriptor (CPD). The descriptor length is 32-bit, with five fields defined according to the specification of Java virtual machine. These fields are, Address Field, Access Modifier, Type Field, Static Flag, Read Only Flag, and Resolved Flag.

2.2 Instruction Set

jHISC is a RISC based core with some object-oriented features enhancement processor. Its Instruction length is 32-bit and supports up to three operands. Each instruction uses 8-bit opcode for defining instruction operation and references data in current LVF with 7 bits, addressing up to 128 registers. In addition, jHISC provides object-oriented manipulation instructions to handle the object-oriented related processing.

In traditional computers, memory-register data transfer instructions allow program to access memory directly which may result in security problems. In jHISC, all data are encapsulated into objects, and each object associates with a pair of memory boundaries (see Section 3). A program needs to pass the bound control checks before it accesses the data and out-of-bound accesses are prohibited. Two instructions “gifld” and “pifld” are introduced in our architecture to perform data transfer operations between memory and register with rigid memory access checking.

There are totally seven groups of instruction defined in jHISC v3, they are: logical instructions, arithmetic instructions, branching instructions, array Instructions, Object-Oriented (OO) instructions and data manipulation instructions. Excluding the float-point and 64-bit operation instructions, jHISC implements 93% bytecodes and 83% OO related bytecodes in hardware directly (Table 1). The rests are executed through software traps, such as “new”, “newarray”.

Table 1. The bytecodes supported by jHISC

Number of bytecodes	226
Number of bytecodes excluding the float-point and 64-bit operations	140
Number of bytecodes supported by the hardware directly	130
Number of bytecodes done by the software traps	10
Number of bytecodes for OO operations	40
Number of OO bytecodes supported by the hardware directly	33
Percentage of bytecodes supported by the hardware directly	93%
Percentage OO bytecodes supported by the hardware directly	83%

3 Security Features

Nowadays computing systems are multi-tasked. Multiple applications are executed on the same machine simultaneously. As more parties are involved in a computing system, it raises some concern of information privacy between different applications.

For object-oriented hardware approach, different objects share the same addressing space. Protection features for object access should be provided by the processor and used by the operating system in order to guarantee security between objects. Unlike traditional processor architecture, object-oriented processor has built-in protection logics to ensure security, in order to minimize execution errors.

Operations in a processor can be roughly categorized into two groups. The first group performs real calculation and data movement. They are used for generating the results of execution. Another group provides the support for control flow. Such operations include branching, subroutine calling and returning. They let a sequential execution machine to change the order of instruction execution according to the flow of program.

The switch of control from one domain to another provides some possible back holes for hacking programs and viruses. In order to secure the system from damages, checking routines are introduced directly to the architecture. In jHISC v3, the secure control transfer mechanism is based on object-oriented programming and the concept of object-oriented programming features like data encapsulation, messaging, etc, is used to restrict access of information from intrusion.

3.1 Strong Type and Structural Memory Access

jHISC v3 is a strong typed system based on the definition of objects. Every data is associated with a type field, which indicates the data type. Data are encapsulated inside an object. All data types are specified upon compilation and they cannot be changed at run-time.

Direct memory accessing operations is prohibited in the jHISC v3 system. Instead of using direct memory accessing operations like load/store, jHISC v3 provides indirect memory accessing operations such as oo.getfield, oo.putfield, array.load and array.store.

3.2 Local Variable Frame and Object Data Structure Out of Bound Checking

Local variables are maintained inside Register Stack Engine (RSE). The RSE contains embedded logics for checking the out of boundary access of local variables. Any out of boundary access to local variables would cause the system a protection violation. This feature removes the potential bugs on the software or corrupted executable programs.

Pair of registers is maintained in the processor for object data structure access. A base register is used to locate the address of the object data and a size register records its size for out of boundary checking. Logics are embedded to do this kind of checking every time the data is accessed through the object-oriented instructions. Since every object has only bounded access to other objects, the sharing of address spaces in object-oriented hardware approach could be achieved with this protection features.

3.3 Intra-method and Inter-method Control Transfer

Intra-method control transfer instructions are defined for conditional statements in programming languages, such as if-then-else, while-loop, for-loop, etc. In jHISC v3, these instructions are based on branching instructions with an additional checking to avoid branching outside of the current method space. A pair of register is used to store the base address and size of current running method. The branch target is an offset to the base address (Fig. 4). The avoidance of branching outside the current method space enables all objects running in the same addressing space.

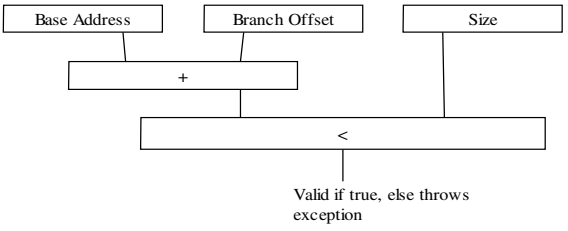


Fig. 4. Branch target checking in jHISC v3

Inter-method control transfer instructions include method invocation and method revocation. Method invocation is a procedure to call a method, while method revocation is a procedure to return from a method. They provide functionality for messaging between objects. Each method in jHISC v3 is defined to have only single entry point of calling. This feature prohibits the code bypassing in the same addressing space (Fig 5).

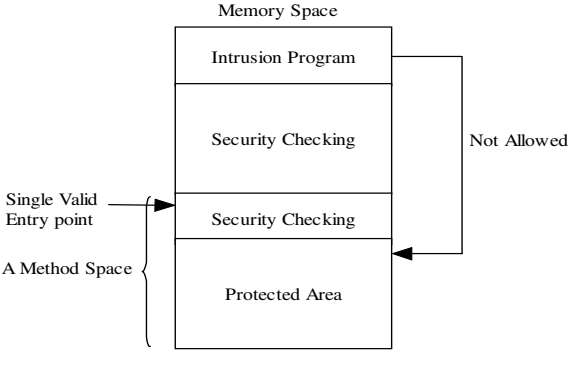


Fig. 5. Single entry point of Method in jHISC v3

4 Co-verification Methodology and Result

The jHISC V3 with 4KB instruction cache and 8KB data cache has already been implemented on a Xilinx Virtex XCV800 FPGA to verify our concept. During implementation, the caches are generated by Xilinx CORE Generator and the whole system costs about 600K equivalent gates in FPGA.

In order to test the compatibility of running Java programs on our architecture, we built a software verification platform to compile all the JVM instructions into jHISC instructions and verify the functionality of the secure architecture through the Just-In-Time compilation of bytecode stream and run it on the FPGA. With the software testing implementation of Just-in-time compilation for jHISC, compatibility of Java program can be assured and tested. With some I/O display operations that have been implemented, the bytecode can be compiled and loaded into the FPGA board through the PCI interface and the required results can be dumped back for display. The linker has also been built to link and demonstrate the executable results that were obtained in our architecture. As shown in Fig. 6, the executable memory map is loaded into the FPGA board through the linker by calling the DimeJavaAPI.

All implemented instructions are tested to verify whether they are operated correctly. The related results are shown on Table 2. Besides, assembly program are written to infringe the system and test whether the jHISC core can detect such infringement and throws an exception to the system. The results are shown on Table 3.

Finally, some testing Java programs are compiled through our JIT compilation and loaded into the system for execution. With the verification of all the data structures and resolutions which are built through the compiler with GUI display (Fig. 7), the compiled code are successfully run in the jHISC core and the results are dumped back through the linker to the Console (see Fig. 8). Table 4 shows results of testing on some simple Java program.

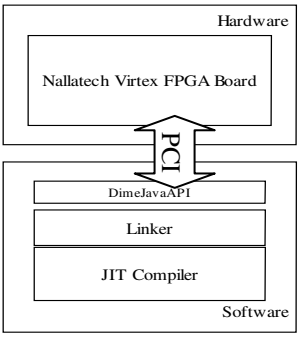


Fig. 6. The Flow of the Co-verification Methodology

Table 2. Result of Instruction Verification

Instruction Group	Result
Logical Instructions	passed
Arithmetic Instructions	passed
Branching Instructions	passed
Array Instructions	passed
Object-Oriented Instructions	passed
Data Manipulation Instructions	passed

Table 3. Result of Protection Features Verification

Tests	Result
Index a register that is out of current LVF	Causes Exception
Modification of a reference type register	Causes Exception
Operation on two different typed (Float and Int) register	Causes Exception
Branch outside the current MCS	Causes Exception
Access an element index that is larger than the array size	Causes Exception
Invoke a non reference type CPD	Causes Exception
Invoke with an index that is larger than CPDTS	Causes Exception
Invoke a non-static method in a static method	Causes Exception
The address of CPB is larger than CDSS	Causes Exception
Invoke a non-method typed object	Causes Exception
Allocate registers (In + Local + out) larger than 32	Causes Exception
Incompatible argument size for invoked method	Causes Exception
Invoke with an index that is larger than CODTS	Causes Exception
Invoke with a non property type COD	Causes Exception
Invoke an unresolved method	Causes Exception
Invoke with a class reference that is not class typed	Causes Exception
Revoke with IVKTYPE = 00b	Causes Exception
Access non-static field in static method	Causes Exception
Access a static field that out of CDS	Causes Exception
Access a field that is out of IDS	Causes Exception
Modify a read only field	Causes Exception
Access an unresolved field	Causes Exception

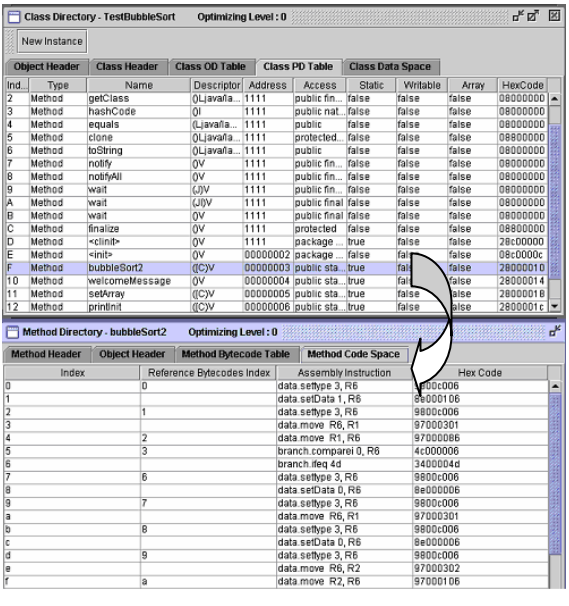


Fig. 7. Class and Method Directory Frame (GUI) displaying information of Class and Method Context which are built from a bubble sort class program and verified through our JIT compiler

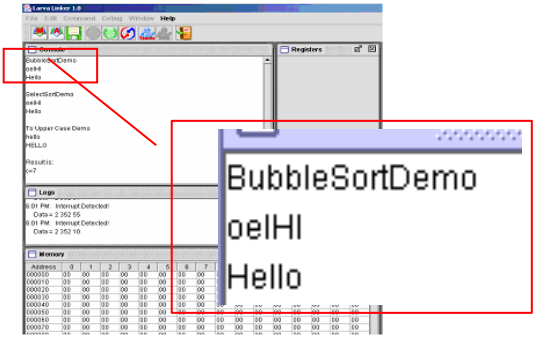


Fig. 8. Executing results of some Java programs

Table 4. Results of testing on some Java programs

Tests	Result
Helloworld – display a 'helloworld' text	Correct
Interactive sorting programs – input a sequence of numbers and sorted the numbers with different sorting	Correct
Perform some arithmetic calculation	Correct
Program to convert all input lower case characters to upper case	Correct
Min/Max – input a sequence of numbers and the minimum and maximum value are displayed	Correct
Message Program – an interactive program that can let user to store message, remove message and read message	Correct

5 Conclusion

In this paper, we have discussed the secure hardware data structure on jHISC, which has been built with FPGA technology, and a software platform to co-verify the compatibility, through Just-In-Time compilation, on some Java class files. Through this hardware/software co-design, some resolution burdens in JVM have been compensated through our novel architecture. Besides, comparing with the traditional system where the security and boundary checking of methods are controlled by software, our approach in putting the security and boundary checking jobs into hardware would move the burden of security checking into some pipeline stages and greatly improve the overall system security and throughput of the processor. Moreover, a secure method invocation can safeguard any error or bad reference produced by erroneous programming. Furthermore, our co-verification process verifies the functionalities of the instructions set, exception handling of our hardware protection features and the execution of some Java program examples. Eventually, Java can be run as a native language in the jHISC architecture and all Object-Oriented and Java's protection features are penetrated from software to hardware in order to increase the overall system security and performance.

Acknowledgements

The work described in this paper was partially supported by a grant from City University of Hong Kong (Strategic Research Grant Project No. 7001548).

References

1. Andrew S. Tanenbaum. Structured Computer Organization, Forth Edition. Prentice Hall, 1999.
2. Fong, A., S., “HISC: A High-level Instruction Set Computer”. In 7th European Simulation Symposium, pages 406-410. The Society for Computer Simulation, Oct 1995.
3. Mok Pak Lun, Richard Li, Anthony Fong. “Method manipulation in an object oriented processor”. ACM SIGARCH Computer Architecture News archive. Volume 31, Issue 4, September 2003.
4. Bill Venners. Inside the Java 2 Virtual Machine. Mc Graw Hill 1999.
5. Tim Lindholm, Frank Yellin The Java(TM) Virtual Machine Specification (2nd Edition). Addison-Wesley Pub Co; 2nd edition,1999
6. Austin Kim, Yang Qian and J. M. Chang, " Designing a Memory System Using a Static Loader For Embedded Java Architectures" , The Second International Workshop on Compiler and Architecture Support for Embedded Systems (CASES'99) , Washington, D.C., October 1-3, 1999
7. Chang, L.-C., Ton, L.-R., Kao, M.-F., Chung, C.-P., “Stack operations folding in Java processors”, Computers and Digital Techniques, IEE Proceedings- Volume 145, Issue 5, Sept. 1998, pp. 333 – 340
8. Kim, A.; Chang, M, “Advanced POC Model-Based Java Instruction Folding Mechanism”, Euromicro Conference, 2000. Proceedings of the 26th Volume 1, 5-7 Sept. 2000, pp. 332 - 338 vol.1
9. Sun Microsystems, Inc. The Java HotSpot performance engine architecture.
10. M. W. El-Kharashi, “Java Microprocessors: Computer Architecture Implications”, IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, Vol. 1, August, 1997, pp. 277-280.
11. D. Mulchandani, “Java for Embedded Systems”, IEEE Internet Computing, June, 1998, pp. 30-39.

Timed Weak Simulation Verification and Its Application to Stepwise Refinement of Real-Time Software

Satoshi Yamane

Graduate School of Natural Science, Kanazawa University,
Kanazawa City, Japan, ZIP/CODE 920-1192
syamane@is.t.kanazawa-u.ac.jp

Abstract. Real-time software runs over real-time operating systems, and guaranteeing qualities are difficult. In this paper, we propose timed weak simulation relation verification and apply it to a refinement design method of real-time software. Moreover, we apply our proposed method to general real-time software scheduled by fixed-priority preemptive policy.

1 Introduction

As real-time software is reactive and concurrent, and its timing conditions are strict, it is difficult to design real-time software [1]. It is important to specify and verify real-time software. In this paper, we propose timed weak simulation relation verification method based on timed automata [2], and apply it to stepwise refinement of real-time software over fixed priority preemptive schedulers [3]. In general, real-time software is designed by dividing it into tasks [1] In this case, it is difficult to design real-time software by the following points: (1)Real-time software consists of many tasks, which concurrently behave. Moreover, tasks interact with external environments. In this situation, it is useful to distinguish between internal events and external events in the sense of process algebra [4]. (2)In real-time software, stepwise refinement is useful, and it is important to automatically verify whether the concrete specification refines the abstract specification.

From the above results, we propose the followings: (1)We use nondeterministic timed automata, which have internal and external events. We construct real-time software by parallel composition of nondeterministic timed automata. (2)We verify whether the concrete specification refines the abstract specification based on a timed weak simulation.

In general, refinement relations such as language inclusion, timed bisimulation and timed strong simulation are useful as follows: (1)We can easily and naturally verify fairness and regularity as acceptance conditions by language inclusion. But if we specify verification properties using nondeterministic timed automata, language inclusion problems are undecidable [2]. On the other hand, R. Alur proposed an event-clock automata, which is a determinizable of timed

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 381–394, 2005.
© IFIP International Federation for Information Processing 2005

automaton [5,6]. But an event-clock automaton is a subclass of a timed automaton, and accepts a finite timed word (though a general timed automaton accepts an infinite timed word). As the determinization of an event-clock automaton causes an exponential blow-up in the number of locations, the verification cost increases. Moreover, we can not verify some deadlock using language inclusion [4,7]. (2)Timed bisimulation relation is useful for verifying a kind of invariant holding between the more concrete specification and the more abstract specification [8]. On the other hand, timed strong simulation relation is useful for verifying stepwise refinement [9]. But when we stepwise develop specifications, we may add exception procedures to the concrete specification, which are not contained in the abstract specification. Both timed bisimulation relation and timed strong simulation relation are not adequate for this reason.

From the above result, we use timed weak simulation relation in order to verify whether the concrete specification refines the abstract specification.

We survey related works as follows: (1)In 1992, Cerans has shown that timed strong and weak bisimulation equivalence problem for timed automata are decidable [8]. But he has not developed bisimulation algorithms. (2)In 1996, Tasiran and his colleagues have developed the verification algorithm of timed strong simulation relation [9]. But they have not developed a timed weak simulation relation. (3)In 1999, Braberman and his colleagues have developed reachability analysis method of preemptive scheduling using timed automata [10]. But they have not developed refinement verification method.

In this paper, we define a timed weak simulation relation, and propose the verification algorithm of timed weak simulation relation. Moreover we apply our proposed method to general real-time software scheduled by fixed-priority preemptive policy. As it is difficult to treat internal behaviors, timed weak simulation verification methods of timed automata have never been developed before now. Timed weak simulation verification is our original work. The paper is organized as follows: In section 2, we define specification method. In section 3, we define timed weak simulation relation verification method and apply it to stepwise refinement of real-time software. In section 4, we present design supprt system and some example. Finally, in section 5, we present conclusions.

2 Specification of Real-Time Software

In this paper, we specify tasks using nondeterministic timed automata with internal and external events, and construct real-time software by parallel composition of tasks.

2.1 Syntax and Semantics of Timed Automata

First we define clock as follows:

Definition 1 (Clock and clock interpretation).
Given a finite set of variables $X = \{x_1, \ldots, x_n\}$*, a valuation is a function* $v : X \to \mathbb{R}^{+0}$*, which assigns a nonnegative real value to each clock variable. We*

define V^X *as the set* $[X \to \mathbb{R}^{+0}]$. $\mathbf{0}$ *denotes the valuation that assigns the value 0 to each* $x \in X$. *For* $\lambda \subseteq X$, $v[\lambda := 0]$ *denotes the valuation that assigns the value 0 to each* $x \in \lambda$ *and agrees with* v *for all clocks in* $X \setminus \lambda$. *Moreover, for every* $t \in \mathbb{R}^+$, $v + t$ *denotes the clock valuation for which all clocks* $x \in X$ *take the value* $v(x) + t$.

■

Definition 2 (Clock constraints).
For a set X *of clock variables, the set* Ψ^X *of clock constraints* ψ *is inductively defined by* $\psi ::= x \sim c \mid \psi_1 \wedge \psi_2$, *where* $\sim \in \{\leq, =, \geq\}$, $c \in \mathbb{N}$. *We write* $v \models \psi$ *if the valuation* v *satisfies the formula* ψ. *For each clock* $x \in X$, $c_x(\Psi^X)$ *denotes the maximal clock constant in* Ψ^X.

■

Definition 3 (Syntax of timed automata).
A timed automaton $\mathbf{G}$ *is a tuple* $\langle S, \bar{s}, \Sigma, X, inv, E\rangle$, *where*

(1) S *is the finite set of locations.*
(2) $\bar{s} \in S$ *is an initial location.*
(3) $\Sigma = EXT \cup INT$ *is the finite set of events, where* EXT *is the finite set of external events,* INT *is the finite set of internal events.*
(4) X *is the finite set of real-valued variables, called clocks.*
(5) $inv : S \to \Psi^X$ *is the invariant function that assigns* Ψ^X *to each location* $s \in S$.
(6) E *is the finite set of edges. Each edge* e *is a tuple* $\langle s, \sigma, \psi, \lambda, s'\rangle = e \in E$ *consisting of the source location* s, *the target location* $s\prime$, *clock constraint* $\psi \in \Psi^X$, *the set* λ *of clocks to be reset,* $\sigma \in \Sigma$ *is an event, where* $\lambda = \emptyset$ *if* $\sigma \in INT$.

■

Next we formally define semantics of timed automata.

Definition 4 (Semantics of timed automata).
A state of $\mathbf{G}$ *is a pair* $\langle s, v\rangle$ *containing the location* $s \in S$ *and the valuation* $v \models inv(s)$. *The set of all states is denoted* Ω. *The initial state is a pair* $\langle \bar{s}, \mathbf{0}\rangle$. *For each state* $\langle s, v\rangle$, *the transition is defined as follows:*

1. *Discrete transitions:*
$$\frac{\langle s, \sigma, \psi, \lambda, s'\rangle \in E,\ v \models \psi,\ v[\lambda := 0] \models inv(s')}{\langle s, v\rangle \xrightarrow{\sigma} \langle s', v'\rangle}\text{-},\ \textit{where } v' = v[\lambda := 0].$$
2. *Timed transitions:*
$$\frac{\delta \in \mathbb{R}^+\ \forall \delta' \leq \delta.\ v + \delta' \models inv(s)\ \textit{implies}\ \langle s, v + \delta\rangle}{\langle s, v\rangle \xrightarrow{\delta} \langle s, v'\rangle}\text{-},\ \textit{where } v' = v + \delta.$$

A run of timed automaton is an infinite sequence as follows:

$$\langle \overline{s}, \mathbf{0}\rangle \xrightarrow{\ell_1} \langle s_1, v_1\rangle \xrightarrow{\ell_2} \langle s_2, v_2\rangle \xrightarrow{\ell_3} \cdots,$$

where $\langle \overline{s}, \mathbf{0}\rangle$ *is an initial state,* $\langle s_i, v_i\rangle \in \Omega$ *is a state,* $\ell_i \in (\Sigma \cup \mathbb{R}^+)$ *is a label.*

■

In this paper, we assume that timed automaton is *nonZeno*. It is easy to verify whether a timed automaton is *nonZeno* or not using HYTECH [11].

2.2 Parallel Composition of Timed Automata

In this paper, we construct real-time software by parallel composition of tasks. We define parallel composition of timed automata as follows:

(1) If the external event of a task is equal to the external event of environments, the task is synchronized with environments by the same external event.
(2) As internal events of tasks are unobservable from environments, internal events of tasks and events of environments are disjoint.

Definition 5 (Parallel composition).
Let be two timed automata $\mathbf{G_1} = \langle S_1, \bar{s}_1, \Sigma_1, X_1, inv_1, E_1 \rangle$ *and* $\mathbf{G_2} = \langle S_2, \bar{s}_2, \Sigma_2, X_2, inv_2, E_2 \rangle$. *The parallel composition of* $\mathbf{G_1}$ *and* $\mathbf{G_2}$ *is the timed automaton* $\mathbf{G} = \langle S, \bar{s}, \Sigma, X, inv, E \rangle$, *where* $\Sigma_1 = EXT_1 \cup INT_1$, $\Sigma_2 = EXT_2 \cup INT_2$. *Here* $INT_1 \cap \Sigma_2 = \Sigma_1 \cap INT_2 = \emptyset$.

(1) $S \subseteq S_1 \times S_2$.
(2) $\bar{s} = (\bar{s}_1, \bar{s}_2)$.
(3) $\Sigma = EXT \cup INT$, *where* $EXT = EXT_1 \cup EXT_2$, $INT = INT_1 \cup INT_2$.
(4) $X = X^A \cup X^B$.
(5) $inv((s_1, s_2)) = inv_1(s_1) \wedge inv_2(s_2)$.
(6) $\langle s, \sigma, \psi, \lambda, s' \rangle \in E$, *where for* $\langle s_1, \sigma, \psi_1, \lambda_1, s_1' \rangle \in E_1$ *and* $\langle s_2, \sigma, \psi_2, \lambda_2, s_2' \rangle \in E_2$, *each element is as follows:*
  (a) When $\sigma \in \Sigma_1 \cap \Sigma_2$, $s = (s_1, s_2), \psi = \psi_1 \wedge \psi_2, \lambda = \lambda_1 \cup \lambda_2, s' = (s_1', s_2')$.
  (b) When $\sigma \in \Sigma_1$ *and* $\sigma \notin \Sigma_2$, $s = (s_1, s_2), \psi = \psi_1, \lambda = \lambda_1, s' = (s_1', s_2)$.
  (c) When $\sigma \in \Sigma_2$ *and* $\sigma \notin \Sigma_1$, $s = (s_1, s_2), \psi = \psi_2, \lambda = \lambda_2, s' = (s_1, s_2')$.

■

2.3 Specification Method

We decide parameters such as priorities and timing constraints by V. Braberman's method [10], which is based on WCRT(Worst Case Response Time) [12].

First we define Worst-Case Response Time as follows:

Definition 6 (Worst-Case Response Time).
If every task j, $j < i$, *has higher priority than task* i, *the worst-case response time* R_i *of task* i *is given as recursive equation* $(i = 1, \ldots, n)$. *The* $(k+1)th$ *worst-case response time* $Ri^{(k+1)}$ *for task* i *is as follows* $(k \geq 0)$ *:*

$$R_i^{(k+1)} = \sum_{j=1}^{i-1} (\lceil \frac{R_i^{(k)}}{T_j} \rceil \times C_j) + C_i,$$

where period T_i, *execution time* C_i, *deadline* D_i *of a periodic task* i. *We can compute* $R_i = lim_{k \to \infty} R_i^{(k)}$ *as* $Ri^{(0)} = C_i$.

■

Using R_i, we can check whether real-time software is schedulable or not as follows:

Real-time software is schedulable if the following condition is satisfied:
For $\forall i$, $R_i \leq D_i$(i=1,..,n) holds true.[12]

Next we specify real-time software using timed automata.

In general, it is not possible to exactly specify preemptive scheduling using timed automata. Therefore, R. Alur and T.A. Henzinger have specified preemptive scheduling using hybrid automata [13]. In this paper, we approximately specify timing constraints by $c_{min} \leq x \leq c_{max}$ using timed automata, where we set c_{min} using timing constraints of edges, and set c_{max} using worst-case response time. Therefore, we can realize the automatic verification of timed weak simulation relation. If we specify real-time software of preemptive scheduling using hybrid automata, it is not possible to automatically verify timed weak simulation [14].

3 Refinement Design Method

It is important to design real-time software by stepwise refinement as real-time software is a complex system. In this case, it is important to verify whether the concrete specification is satisfied by the abstract one or not. First we define a timed weak simulation. Next we define the verification method of a timed weak simulation. Finally we explain the stepwise refinement design method of real-time software.

3.1 A Timed Weak Simulation

First we define observable transitions as follows:

Definition 7 (Observable transitions).
For each state $\langle s, v\rangle \in \Omega$ of timed automaton $\mathbf{G} = \langle S, \bar{s}, \Sigma, X, inv, E\rangle$, observable transitions are defined as follows:

Here Let $\stackrel{\epsilon}{\Longrightarrow}$ if and only if $(\stackrel{\tau}{\rightarrow})^*$, where $\Sigma = EXT \cup INT$, $\tau \in INT$.

(1) For an external event $\sigma \in EXT$,
Define $\langle s, v\rangle \stackrel{\sigma}{\Longrightarrow} \langle s', v'\rangle$ as $\langle s, v\rangle \stackrel{\epsilon}{\Longrightarrow}\stackrel{\sigma}{\rightarrow}\stackrel{\epsilon}{\Longrightarrow} \langle s', v'\rangle$.

(2) For delay $\delta_1, \ldots \delta_k, \delta \in \mathbb{R}^+$,
Define $\langle s, v\rangle \stackrel{\delta}{\Longrightarrow} \langle s', v'\rangle$ as $\langle s, v\rangle \stackrel{\epsilon}{\Longrightarrow}\stackrel{\delta_1}{\rightarrow} \cdots \stackrel{\epsilon}{\Longrightarrow}\stackrel{\delta_k}{\rightarrow}\stackrel{\epsilon}{\Longrightarrow} \langle s', v'\rangle$ $(k \geq 0)$.

In this case, a state $\langle s, v\rangle$ is called stable from environments, and we denote $\mathbf{wait}(\langle s, v\rangle)$, *where $\delta = \delta_1 + \cdots + \delta_k$*

■

Definition 8 (Timed weak simulation).
Let $\mathbf{G_1} = \langle S_1, \bar{s}_1, \Sigma_1, X_1, inv_1, E_1\rangle$ and $\mathbf{G_2} = \langle S_2, \bar{s}_2, \Sigma_2, X_2, inv_2, E_2\rangle$ be two timed automata. A timed weak simulation relation from $\mathbf{G_1}$ to $\mathbf{G_2}$ is a binary relation $Sim \subseteq \Omega_1 \times \Omega_2$ if the following three conditions are satisfied. Moreover we denote $\mathbf{G_1} \preceq \mathbf{G_2}$ if there exists a timed weak simulation relation, where

Ω_1 is the set of $\langle s_1, v_1\rangle$, Ω_2 is the set of $\langle s_2, v_2\rangle$, $\Sigma_1 = EXT_1 \cup INT_1$, $\Sigma_2 = EXT_2 \cup INT_2$, $s_1, s_1' \in S_1$, $v_1, v_1' \in V^{X_1}$, $s_2, s_2' \in S_2$, $v_2, v_2' \in V^{X_2}$.

(1) External event condition:
$EXT_1 \subseteq EXT_2$.

(2) Simulation condition:
For every $(\langle s_1, v_1\rangle, \langle s_2, v_2\rangle) \in Sim$ and for every $\theta \in (EXT_1 \cup \mathbb{R}^+)$,
if $\langle s_1, v_1\rangle \stackrel{\theta}{\Longrightarrow} \langle s_1', v_1'\rangle$ then there exists $\langle s_2', v_2'\rangle$ such that
$\langle s_2, v_2\rangle \stackrel{\theta}{\Longrightarrow} \langle s_2', v_2'\rangle$ and $(\langle s_1', v_1'\rangle, \langle s_2', v_2'\rangle) \in Sim$.

(3) Initial condition:
$(\langle \bar{s}_1, \mathbf{0}\rangle, \langle \bar{s}_2, \mathbf{0}\rangle) \in Sim$.

■

3.2 Verification Method of a Timed Weak Simulation

We achieve a timed weak simulation by converting this check to a finite check on the finitely many equivalence classes, which is called a region weak simulation relation. First we define a region graph, and next define a region weak simulation, and the conversion technique. Finally we show the verification algorithm of a timed weak simulation.

Region graph. Since the number of states is infinite, we cannot possibly build a finite automaton. But if two states with the same location agree on the integral parts of all clock values, and also the ordering of the fractional parts of all clock values, then the runs starting from the two states are very similar. From the above facts, we can construct region graphs, which are finite quotient structures by equivalence relations [2].

First we define equivalence relations of clock values.

Definition 9 (Equivalence relations of clock values).
Let V^X be the set of clock values, and Ψ^X be the set of clock constraints. For any $t \in \mathbb{R}^{+0}$, $\lfloor t \rfloor$ denotes the integral part of t, and $fract(t)$ denotes the fractional part of t. For $v, v' \in V^X$, v and v' are equivalent iff the following three conditions are satisfied. We denote $v \cong v'$.

(1) For clock $x \in X$, $\lfloor v(x) \rfloor$ and $\lfloor v'(x) \rfloor$ are the same, or both $v(x)$ and $v'(x)$ are greater than $c_x(\Psi^X)$.

(2) For all $x, y \in X$ with $v(x) \leq c_x(\Psi^X)$ and $v(y) \leq c_y(\Psi^X)$, $fract(v(x)) \leq fract(v(y))$ iff $fract(v'(x)) \leq fract(v'(y))$.

(3) For $x \in X$ with $v(x) \leq c_x(\Psi^X)$, $fract(v(x)) = 0$ iff $fract(v'(x)) = 0$.

■

We use $[v]$ to denote the clock region to which v belongs.

Next we define the successor of equivalence classes.

Definition 10 (Successor of equivalence classes).
Let α and β be distinct clock equivalence classes of V^X and Ψ^X. For each $v \in \alpha$ and any $\delta \in \mathbb{R}^+$, we define the successor of equivalence classes:

(1) We denote $\beta = \mathbf{succ_0}(\alpha)$ *iff there exists* δ *such that* $\alpha = \beta$ *and* $\delta \in \beta$.
(2) We denote $\beta = \mathbf{succ_1}(\alpha)$ *iff there exists* $\delta' \leq \delta$ *such that* $\alpha \neq \beta$ *and* $v + \delta' \in \alpha \cup \beta$ *and* $v + \delta \in \beta$.

■

Region is denoted by $\langle s, [v] \rangle$, or, $\langle s, \alpha \rangle$.
Next we define region graph of timed automaton as follows:

Definition 11 (Region graph).
For a timed automaton $\mathbf{G} = \langle S, \bar{s}, \Sigma, X, inv, E \rangle$, *the corresponding region graph* $\mathbf{R}(\mathbf{G}) = \langle Q, \bar{q}, L, N \rangle$ *consists of four tuples:*

(1) the finite set of states Q.
(2) the initial state $\bar{q} \in Q$, *where* $\bar{q} = \langle \bar{s}, [\mathbf{0}] \rangle$.
(3) the finite set of labels $L = \Sigma \cup SUCC$, *where* $SUCC$ *is the set of labels, which represent successor relations of equivalence classes.*
(4) a set of transition relations $N \subseteq Q \times L \times Q$.
For any $\langle s, \alpha \rangle$, *a set of transition relations are defined as follows:*

(a) If there exists $\langle s, v \rangle \xrightarrow{\sigma} \langle s', v' \rangle$ *such that* $v' \in \beta$ *for each* $v \in \alpha$, *it is possible to transit to* $\langle s', \beta \rangle$ *by an event, and we denote* $\langle s, \alpha \rangle \xrightarrow{\sigma} \langle s', \beta \rangle$.

(b) If there exists $\langle s, v \rangle \xrightarrow{\delta} \langle s, v' \rangle$ *such that* $\beta = \mathbf{succ_i}(\alpha)$ *and* $v' \in \beta$ *for each* $v \in \alpha$, *it is possible to transit to* $\langle s, \beta \rangle$ *by a time delay, and we denote* $\langle s, \alpha \rangle \overset{succ_i}{\rightarrow} \langle s, \beta \rangle$ $(i = 0, 1)$, *where* $succ_i \in SUCC$.

■

Region Weak Simulation Relation. We will show that the problem of checking the existence of a timed weak simulation relation is decidable. We achieve this by converting this check to a finite check on the finitely many equivalence classes of an equivalence relation (what we call region weak simulation relation) defined on parallel composition of timed automata. We define a region weak simulation relation on parallel composition of timed automata from the following reasons:

(1) We can construct all the pairs of $(\langle s_1, v_1 \rangle, \langle s_2, v_2 \rangle) \in Sim$ as $\langle (s_1, s_2), v_{12} \rangle$ of parallel compostion of timed automata, where s_1 and v_1 are the node and clock value of timed automaton 1, s_2 and v_2 are the node and clock value of timed automaton 2, v_{12} is the clock values of parallel composition of timed automaton 1 and 2. Therefore a set of state pairs Sim, which represent region weak simulation relation, is a subset of a set of states of product automaton (parallel composition of timed automata) $\Omega_{\mathbf{G_1} \| \mathbf{G_2}}$. Namely, $Sim \subseteq \Omega_{\mathbf{G_1} \| \mathbf{G_2}}$.
(2) As it is easy to trace the relation between v_1 and v_2, we trace it by v_{12} of product automaton.
(3) If one timed automaton has $\langle s_1, v_1 \rangle \xrightarrow{\delta} \langle s_1, v_1' \rangle$ and another timed automaton has $\langle s_2, v_2 \rangle \xrightarrow{\delta} \langle s_2, v_2' \rangle$, product automaton has $\langle (s_1, s_2), v_{12} \rangle \xrightarrow{\delta} \langle (s_1, s_2), v_{12}' \rangle$. Therefore we easily represent two timed automata by product timed automaton.

Definition 12 ($\mathbf{R}(\mathbf{G_1} \parallel \mathbf{G_2})$).
We construct product timed automaton $\mathbf{G_1} \parallel \mathbf{G_2}$ from $\mathbf{G_1}$ and $\mathbf{G_2}$ by parallel composition, where $EXT_1 \subseteq EXT_2$. We define the region of region graph $\mathbf{R}(\mathbf{G_1} \parallel \mathbf{G_2})$ as $\langle (s_1, s_2), \alpha \rangle$, where $s_1 \in S_1$, $s_2 \in S_2$, α is the equivalence class of $V^{X_1 \cup X_2}$ and $\Psi^{X_1 \cup X_2}$. Let $Q_{\mathbf{G_1} \parallel \mathbf{G_2}}$ be the set of equivalence classes on $\mathbf{G_1} \parallel \mathbf{G_2}$, where $\langle (s_1, s_2), \alpha \rangle \in Q_{\mathbf{G_1} \parallel \mathbf{G_2}}$. With $\mathbf{R}(\langle s_1, v_1 \rangle, \langle s_2, v_2 \rangle)$, denote the equivalence class $\langle (s_1, s_2), \alpha \rangle$ that the state $(\langle s_1, v_1 \rangle, \langle s_2, v_2 \rangle) \in \Omega_{\mathbf{G_1} \parallel \mathbf{G_2}}$ belongs to.

For any region $\langle (s_1, s_2), \alpha \rangle$, observable transitions are as follows:

(1) For an external event $\sigma \in EXT$,
Define $\langle (s_1, s_2), \alpha \rangle \overset{\sigma}{\Longrightarrow} \langle (s_1', s_2'), \beta \rangle$ as $\langle (s_1, s_2), \alpha \rangle \overset{\epsilon}{\Longrightarrow} \overset{\sigma}{\rightarrow} \overset{\epsilon}{\Longrightarrow} \langle (s_1', s_2'), \beta \rangle$.

(2) For $succ_i \in SUCC$,
Define $\langle (s_1, s_2), \alpha \rangle \overset{succ_i}{\Longrightarrow} \langle (s_1', s_2'), \beta \rangle$ $(i = 0, 1)$ as $\langle (s_1, s_2), \alpha \rangle \overset{\epsilon}{\Longrightarrow} \overset{succ_i}{\rightarrow} \overset{\epsilon}{\Longrightarrow} \langle (s_1', s_2'), \beta \rangle$.

■

Next we define a region weak simulation relation on region graph $\mathbf{R}(\mathbf{G_1} \parallel \mathbf{G_2})$.

Definition 13 (Region weak simulation relation).
We say that $\chi \subseteq Q_{\mathbf{G_1} \parallel \mathbf{G_2}}$ is a region weak simulation from $\mathbf{G_1}$ to $\mathbf{G_2}$ iff for each $\mathbf{R}(\langle s_1, v_1 \rangle, \langle s_2, v_2 \rangle) \in \chi$, the following three conditions are satisfied.

(1) For every $\sigma \in EXT$, if $\langle s_1, v_1 \rangle \overset{\sigma}{\Longrightarrow} \langle s_1', v_1' \rangle$, $\langle s_2, v_2 \rangle \overset{\sigma}{\Longrightarrow} \langle s_2', v_2' \rangle$ and $\mathbf{R}(\langle s_1, v_1 \rangle, \langle s_2, v_2 \rangle) \overset{\sigma}{\Longrightarrow} \mathbf{R}(\langle s_1', v_1' \rangle, \langle s_2', v_2' \rangle)$ such that $\mathbf{R}(\langle s_1', v_1' \rangle, \langle s_2', v_2' \rangle) \in \chi$.

(2) If $\mathbf{wait}(\langle s_1, v_1 \rangle)$, for any $succ_i \in SUCC$, $\mathbf{R}(\langle s_1, v_1 \rangle, \langle s_2, v_2 \rangle) \overset{succ_i}{\Longrightarrow} \mathbf{R}(\langle s_1', v_1' \rangle, \langle s_2', v_2' \rangle)$ such that $\mathbf{R}(\langle s_1', v_1' \rangle, \langle s_2', v_2' \rangle) \in \chi$ $(i = 0, 1)$.

(3) $\mathbf{R}(\langle \bar{s}_1, \mathbf{0} \rangle, \langle \bar{s}_2, \mathbf{0} \rangle) \in \chi$.

■

Theorem 1 (Timed weak simulation and region weak simulation).
For $\mathbf{R}(\langle s_1, v_1 \rangle, \langle s_2, v_2 \rangle) \in \chi$, let $\mathbf{R}_\chi = \{(\langle s_1, v_1 \rangle, \langle s_2, v_2 \rangle) | \mathbf{R}(\langle s_1, v_1 \rangle, \langle s_2, v_2 \rangle) \in \chi\}$. $\mathbf{R}_\chi$ is a weak timed simulation relation from $\mathbf{G_1}$ to $\mathbf{G_2}$ iff χ is a region weak simulation relation from $\mathbf{G_1}$ to $\mathbf{G_2}$.

Proof 1. *We prove it by dividing it into two cases.*

(I) To prove that if $\mathbf{R}_\chi$ is a weak timed simulation relation from $\mathbf{G_1}$ to $\mathbf{G_2}$, χ is a region weak simulation relation from $\mathbf{G_1}$ to $\mathbf{G_2}$:
Assuming that $\mathbf{R}_\chi$ is a weak timed simulation relation from $\mathbf{G_1}$ to $\mathbf{G_2}$. From the definition, we can directly prove χ is a region weak simulation relation from $\mathbf{G_1}$ to $\mathbf{G_2}$.

(II) To prove that if χ is a region weak simulation relation from $\mathbf{G_1}$ to $\mathbf{G_2}$, $\mathbf{R}_\chi$ is a weak timed simulation relation from $\mathbf{G_1}$ to $\mathbf{G_2}$:
Assuming that χ is a region weak simulation relation from $\mathbf{G_1}$ to $\mathbf{G_2}$. For some $\theta \in (EXT \cup \mathbb{R}^+)$, $(\langle s_1, v_1 \rangle, \langle s_2, v_2 \rangle) \in \mathbf{R}_\chi$ and

$\langle s_1, v_1\rangle \stackrel{\theta}{\Longrightarrow} \langle s_1', v_1'\rangle$.

We need to show that there exists $\langle s_2', v_2'\rangle$ *such that*

$$\langle s_2, v_2\rangle \stackrel{\theta}{\Longrightarrow} \langle s_2', v_2'\rangle \text{ and } (\langle s_1', v_1'\rangle, \langle s_2', v_2'\rangle) \in \mathbf{R}_\chi.$$

(1) When θ *is* $\sigma \in EXT$*:*

From the definition of $\mathbf{R}(\mathbf{G_1} \parallel \mathbf{G_2})$*, there exists* $\langle s_2', v_2'\rangle$ *such that*

$$\langle s_2, v_2\rangle \stackrel{\sigma}{\Longrightarrow} \langle s_2', v_2'\rangle \text{ and } (\langle s_1', v_1'\rangle, \langle s_2', v_2'\rangle) \in \mathbf{R}_\chi.$$

(2) When θ *is* $\delta \in \mathbb{R}^+$*:*

Let be $\mathbf{R}(\langle s_1, v_1\rangle, \langle s_2, v_2\rangle) = \langle (s_1, s_2), [v]\rangle$ *and* $\mathbf{R}(\langle s_1', v_1'\rangle, \langle s_2', v_2'\rangle) = \langle (s_1', s_2'), [v']\rangle$.

First we define equivalence classes $\alpha_0, \ldots, \alpha_{k+1}$ *corresponding to* v *as follows:*

$$\begin{cases} \alpha_0 = [v] \\ \alpha_{i+1} = \mathbf{succ_1}(\alpha_i)\ (0 \le i \le k) \end{cases}$$

Next we define real values $\delta_1, \ldots, \delta_k, \delta'$ *and clock values* $v^{(0)}, \ldots, v^{(k)}$ *corresponding to* α_i *as follows:*

$$\begin{cases} v^{(0)} = v \\ v^{(i+1)} = v^{(i)} + \delta_{i+1} \in \alpha_{i+1}\ (0 \le i < k) \\ v' = v^{(i)} + \delta' \in \alpha_i \cup \alpha_{i+1} \quad (i = k) \end{cases}$$

As there exists k *such that* $\delta = \delta_1 + \cdots + \delta_k + \delta'$*, there are two cases*$(k \ge 0)$.

(a) $k = 0$*:*

In this case, as $v' \in \alpha_0 \cup \alpha_1$ *by* $\delta = \delta'$*, the transition is as follows:*

$$\langle s_1, s_2, \alpha_0\rangle \stackrel{succ_i}{\Longrightarrow} \langle s_1', s_2', \alpha_i\rangle$$

Therefore clearly, there exists $\langle s_2', v_2'\rangle$ *such that*

$\langle s_2, v_2\rangle \stackrel{\delta}{\Longrightarrow} \langle s_2', v_2'\rangle$ *and* $(\langle s_1', v_1'\rangle, \langle s_2', v_2'\rangle) \in \mathbf{R}_\chi\ (i = 0, 1)$.

(b) $k > 0$*:*

We consider the following regions as $\langle s_1^{(i)}, s_2^{(i)}, v_i\rangle \stackrel{\delta_{i+1}}{\Longrightarrow} \langle s_1^{(i+1)}, s_2^{(i+1)}, v_{i+1}\rangle$ *of* $\mathbf{G_1} \parallel \mathbf{G_2}\ (0 \le i \le k)$.

(i) $\langle s_1^{(0)}, s_2^{(0)}, \alpha_0\rangle = \langle s_1, s_2, [v]\rangle$.

(ii) $\langle s_1^{(i)}, s_2^{(i)}, \alpha_i\rangle \stackrel{succ_1}{\Longrightarrow} \langle s_1^{(i+1)}, s_2^{(i+1)}, \alpha_{i+1}\rangle$.

(iii) $\langle s_1^{(i)}, s_2^{(i)}, \alpha_i\rangle \in \chi$.

$\langle s_1^{(k)}, s_2^{(k)}, \alpha_k\rangle \in \chi$ *is inductively derived from* $\langle s_1, s_2, [v]\rangle$*. As* $v' \in \alpha_k \cup \alpha_{k+1}$ *by* $\delta = \delta_1 + \cdots + \delta_k + \delta'$*, we get the following transition:*

$$\langle s_1^{(k)}, s_2^{(k)}, \alpha_k\rangle \stackrel{succ_j}{\Longrightarrow} \langle s_1', s_2', \alpha_{k+j}\rangle.$$

Therefore clearly there exists $\langle s_2', v_2'\rangle$ *such that*

$$\langle s_2, v_2\rangle \stackrel{\delta}{\Longrightarrow} \langle s_2', v_2'\rangle \text{ and } (\langle s_1', v_1'\rangle, \langle s_2', v_2'\rangle) \in \mathbf{R}_\chi\ (j = 0, 1).$$

■

Verification algorithm of a timed weak simulation. We define the verification algorithm of a timed weak simulation as follows:

Definition 14 (Verification algorithm of a timed weak simulation). *For the concrete specification* $\mathbf{G_L}$ *and the abstract specification* $\mathbf{G_H}$*, we construct* $\mathbf{G_L} \parallel \mathbf{G_H}$*, and the region graph* $\mathbf{R}(\mathbf{G_L} \parallel \mathbf{G_H})$*. In this case, we define the verification algorithm in order to verify whether there exists a timed weak simulation from* $\mathbf{G_L}$ *to* $\mathbf{G_H}$*. Basically first we define* $\chi^{(0)}$*, and inductively compute* $\chi^{(k+1)}$ *from* $\chi^{(k)}$*. First if* $EXT_1 \not\subseteq EXT_2$*, there does not exists a timed weak simulation relation. If* $EXT_1 \subseteq EXT_2$*, we compute the followings:*

(1) First we compute the relation $\chi^{(0)} = Q_{\mathbf{G_L} \parallel \mathbf{G_H}}$*, and initialize* k *by* $k := 0$*.*

(2) Next we inductively compute $\chi^{(k+1)}$ *from* $\chi^{(k)}$ *by repeating the following procedures* $(k \geq 0)$*.*

(a) $\chi^{(k+1)} = \emptyset$*.*

(b) If $\mathbf{R}(\langle s_1, v_1\rangle, \langle s_2, v_2\rangle) \in \chi^{(k)}$*, we set* $\chi^{(k+1)} = \chi^{(k+1)} \cup \{\mathbf{R}(\langle s_1, v_1\rangle, \langle s_2, v_2\rangle)\}$ *when the two following conditions are satisfied.*

(i) For every $\sigma \in EXT$*, if there exists* $\langle s_1', v_1'\rangle$ *such that*

$$\langle s_1, v_1\rangle \stackrel{\sigma}{\Longrightarrow} \langle s_1', v_1'\rangle,\ \langle s_2, v_2\rangle \stackrel{\sigma}{\Longrightarrow} \langle s_2', v_2'\rangle,$$

$$\mathbf{R}(\langle s_1, v_1\rangle, \langle s_2, v_2\rangle) \stackrel{\sigma}{\Longrightarrow} \mathbf{R}(\langle s_1', v_1'\rangle, \langle s_2', v_2'\rangle), \text{ and}$$

$$\mathbf{R}(\langle s_1', v_1'\rangle, \langle s_2', v_2'\rangle) \in \chi^{(k)}.$$

(ii) If $\mathbf{wait}(\langle s_1, v_1\rangle)$*, for every* $succ_i \in SUCC$*,*

$$\mathbf{R}(\langle s_1, v_1\rangle, \langle s_2, v_2\rangle) \stackrel{succ_i}{\Longrightarrow} \mathbf{R}(\langle s_1', v_1'\rangle, \langle s_2', v_2'\rangle)$$

and $\mathbf{R}(\langle s_1', v_1'\rangle, \langle s_2', v_2'\rangle) \in \chi^{(k)}$ $(i = 0, 1)$*.*

(c) If $\chi^{(k+1)} = \chi^{(k)}$*, go to (3). If* $\chi^{(k+1)} \neq \chi^{(k)}$*, set* $k := k + 1$*, and return (2)(a).*

(3) If $\chi^{(k+1)}$ *includes* $\mathbf{R}(\langle \bar{s}_1, \mathbf{0}\rangle, \langle \bar{s}_2, \mathbf{0}\rangle)$*, we decide* χ *is a timed weak simulation relation. If not so, we decide* χ *is not a timed weak simulation relation.*

■

As this algorithm can be formalized as the greatest fixpoint computation, the algorithm terminates.

3.3 Stepwise Refinement Design Method

We represent both the abstract specification and the concrete one by nondeterministic timed automata, and verify the consistencies between them by a timed weak simulation relation.

Definition 15 (Stepwise refinement design method). *The stepwise refinement design method of real-time software consists of the following procedures.*

(1) First we decide task priorities and parameters by WCRT, and specify $\mathbf{TASK_i^{(0)}}$ *by timed automata* $(i = 1, \ldots, n)$*, where* n *is a number of tasks and* (0) *is the level of refinement.*

(2) Next we refine $\mathbf{TASK_i^{(k)}}$ *into* $\mathbf{TASK_i^{(k+1)}}$*, and specify* $\mathbf{TASK_i^{(k+1)}}$ *by timed automata* ($k \geq 0$).

(3) Finally, we construct $\mathbf{SOFT^{(k)}} = \mathbf{TASK_1^{(k)}} \parallel \cdots \parallel \mathbf{TASK_n^{(k)}}$ *and* $\mathbf{SOFT^{(k+1)}} = \mathbf{TASK_1^{(k+1)}} \parallel \cdots \parallel \mathbf{TASK_n^{(k+1)}}$*. We revise* $\mathbf{SOFT^{(k+1)}}$ *until there exists a timed weak simulation relation from* $\mathbf{SOFT^{(k+1)}}$ *to* $\mathbf{SOFT^{(k)}}$*.*

(4) We repeat the above step (1)-(3), and specify the final one.

■

4 Example of Refinement Design

We show our proposed method effective by a plant system. We have implemented our proposed method as design support system. The design support system is implemented by C++ language(7000 lines) on Sun Blade1000(CPU UltraSPARC-III 900MHz, Main memory 1GB). The plant system is an embedded real-time system, and behaves by fixed priority preemptive scheduling. The plant system samples data from *Sensor*, and controls $Valve$, and outputs information on *Display*. If *Sensor* is in bad condition, this information is announced to administrator by *Lamp*. Even if *Sensor* is in bad condition, the plant system continue to behave by control data. The real-time software of the plant system consists of two periodic tasks such as $SAMPLER$ and $MANAGER$. The two tasks communicate with each other by $QUEUE$. In this paper, we think the plant system is implemented by rate monotonic scheduling. Therefore this rate monotonic scheduling algorithm assigns priorities to tasks based on their periods: the shorter the period, the higher the priority. Moreover, we assume $D_i = T_i$ ($i = 1, 2$). We compute WCRT(Worst Case Response Time) [12]. From the results, we can determine two tasks are schedulable. The parameters are shown as follows: $SAMPLER$ has parameters such as $T_i = 20$, $C_i = 6$, $B_i = 1$, $R_i = 7$. $MANAGER$ has parameters such as $T_i = 30$, $C_i = 10$, $B_i = 0$, $R_i = 16$.

4.1 Specification of the Plant System

We design the plant system, and specify it by timed automata. External events are classified into three types such as system call($start_i$, end_i, $push_queue$, pop_queue), task control($dispatch_i$) and API(APplication Interface)($read_sensor$, $write_lamp$, $write_display$, $write_valve$). Next we show the shared resource **QUEUE** in Figure 1.

Next we design the abstract specifications $\mathbf{SAMPLER^{(0)}}$ and $\mathbf{MANAGER^{(0)}}$, and specify them by timed automata, and show them in Figure 2 and 3. In abstract specification, we specify only external events and abstract behaviors. For example, we specify $\mathbf{SAMPLER^{(0)}}$ by nondeterministically behaving $write_lamp$. Finally we design the concrete specifications $\mathbf{SAMPLER^{(1)}}$ and $\mathbf{MANAGER^{(1)}}$, and show them in Figure 4 and 5. We refine the abstract specification into the concrete one by adding internal behaviors, transforming nondeterministic behaviors into deterministic behaviors

and specifying detailed timing constraints. For example, **SAMPLER**$^{(1)}$ executes *read_sensor* once or twice, and if it fails, **SAMPLER**$^{(1)}$ executes *make_alternate_data*. As the deviation of control data is larger the processing of *write_valve* takes longer time, in **MANAGER**$^{(1)}$, *deviation_Small* takes 1, *deviation_Medium* takes 2 and *deviation_Large* takes 3.

4.2 Verification Experiment

We verify whether **SOFT**$^{(1)} \preceq$ **SOFT**$^{(0)}$ holds true or not using our design support system, where the abstract specification is defined as **SOFT**$^{(0)}$ =

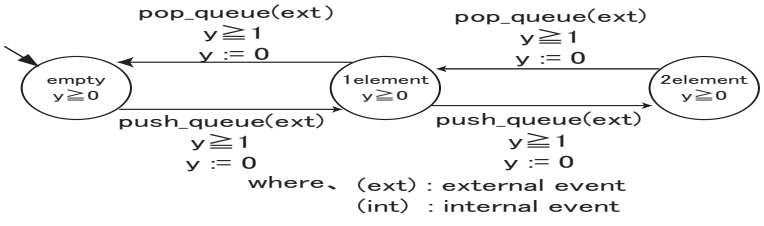


Fig. 1. Specification of the shared resource **QUEUE**

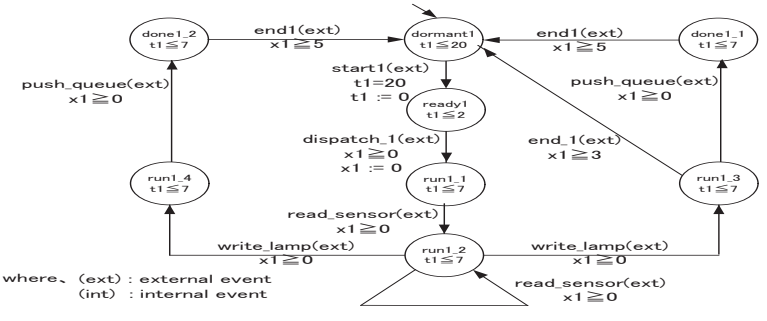


Fig. 2. Specification of the periodic task **SAMPLER**$^{(0)}$

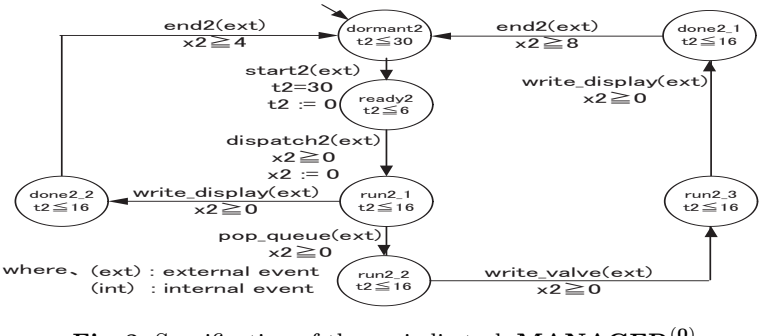


Fig. 3. Specification of the periodic task **MANAGER**$^{(0)}$

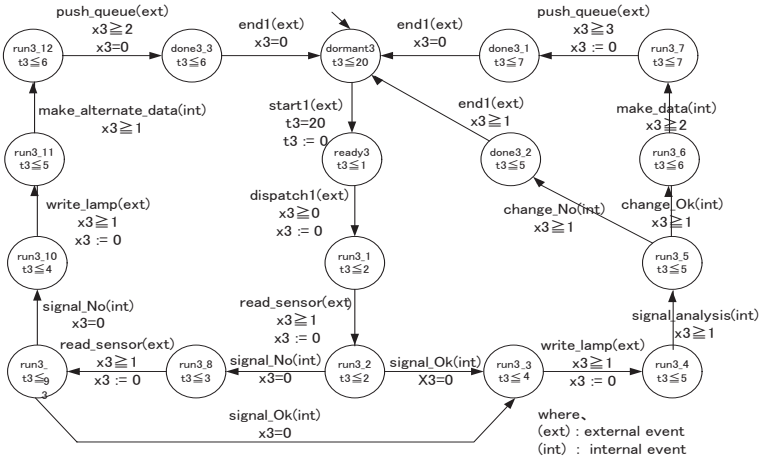


Fig. 4. Specification of the periodic task $\mathbf{SAMPLER^{(1)}}$

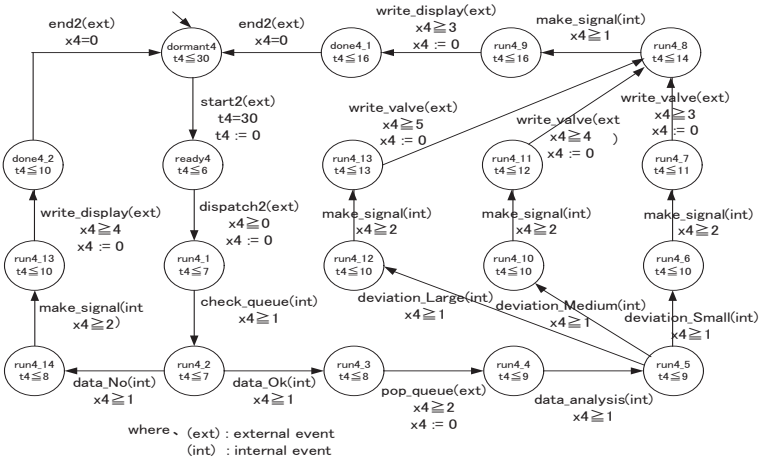


Fig. 5. Specification of the periodic task $\mathbf{MANAGER^{(1)}}$

$\mathbf{SAMPLER^{(0)}} \parallel \mathbf{MANAGER^{(0)}} \parallel \mathbf{QUEUE}$, the concrete specification is defined as $\mathbf{SOFT^{(1)}} = \mathbf{SAMPLER^{(1)}} \parallel \mathbf{MANAGER^{(1)}} \parallel \mathbf{QUEUE}$. We measured required memory=325744kB and execution time=62.30min using ps command. In region graph $\mathbf{R}(\mathbf{SOFT^{(1)}} \parallel \mathbf{SOFT^{(0)}})$, the number of the equivalence classes is 62367, the number of regions of $Q_{\mathbf{SOFT^{(1)}} \parallel \mathbf{SOFT^{(0)}}}$ is 1645498 and the number of members of region weak simulation χ is 1408290. The members of the set χ, which are contained in the region weak simulation relation. As an initial region is contained in χ, $\mathbf{SOFT^{(1)}} \preceq \mathbf{SOFT^{(0)}}$ is satisfied. Namely, there exists a timed weak simulation from the concrete specification to the abstract one. Though the verification cost is large, we can specify both the abstract and

the concrete specifications of real-time software using timed automata, and automatically verify whether there exists a timed weak simulation from the concrete specification to the abstract one.

5 Conclusion

In this paper, we have proposed the following stepwise refinement design methodology of real-time software : First as we extend general timed automata by distinguishing between internal and external events, this timed automata are suitable for stepwise specifying and designing real-time software. Next we automatically verify whether there exists a timed weak simulation relation from the concrete specification to the abstract one. To the best of our knowledge, a timed weak simulation verification methods of timed automata have never been developed before now.

References

1. J.W.S. Liu. : Real-Time System, Prentice-Hall (2000).
2. R. Alur, D.L. Dill. : A theory of timed automatoa, Theoretical Computer Science, Vol. 126, pp.183-235 (1994).
3. Giorgio C. Buttazzo. : Hard Real-Time Computing Systems, Kluwer Academic Publishers (1987).
4. R. Milner. : Operational and Algebraic Semantics of Concurrent Processes. Handbook of Theoretical Computer Science, MIT Press, pp. 1201-1242 (1990).
5. R. Alur, L. Fix, T.A. Henzinger. : Event-Clock Automata: A Determinizable Class of Timed Automata, in Proc. of CAV'94, LNCS 818, pp.1-13 (1994).
6. C. Dima. : Removing slient transitions from event-clock automata, in Proc. of CITTI 2000, pp 75-81 (2000).
7. A. Pnueli. : Linear and Branching Structures in the Semantics and Logics of Reactive Systems. ICALP 1985, volume 194 of LNCS, pp. 15-32 (1985).
8. K. Cerans. : Decidability of bisimulation equivalences for parallel timer processes. LNCS 663, pp.269-300, Springer Verlag (1992).
9. S.Tasiran, R.Alur, R.P.Kurshan, and R.K.Brayton. : Verifying Abstractions of Timed Systems. LNCS 1119, pp.546-562, Springer Verlag (1996).
10. V.Braberman, M.Felder. : Verification of Real-Time Designs: Combining Scheduling Theory with Automatic Formal Verification, LNCS 1687, pp.494-510(1999)
11. T. Henzinger, X. Nicollin, J. Sifakis, S. Yovine. : Symbolic model checking for real-time systems, Information and Computation 111, pp.193–244(1994).
12. M.Joseph. : Real-Time System Specification, Verification and Analysis. Prentice Hall (1996).
13. R. Alur, T.A. Henzinger, P.-H. Ho. : Automatic symbolic verification of embedded systems. IEEE Trans. on Software Engineering 22(3), pp.181-201(1996).
14. T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya. : What's decidable about hybrid automata? Journal of Computer and System Sciences 57, pp.94–124(1998).

Checking Component-Based Embedded Software Designs for Scenario-Based Timing Specifications*

Jun Hu, Xiaofeng Yu, Yan Zhang, Tian Zhang,
Xuandong Li, and Guoliang Zheng

State Key Laboratory of Novel Software Technology,
Department of Computer Science and Technology,
Nanjing University, Nanjing, Jiangsu,
P.R. China 210093
hujun@seg.nju.edu.cn

Abstract. In this paper, for real-time embedded software we consider the problem of checking component-based designs for scenario-based timing specifications. By adding time intervals on the actions, we extend the interface automata for modelling real-time systems. The component-based designs are modelled by real-time interface automaton networks, which includes a set of real-time interface automata synchronized by shared actions, and the scenario-based timing specifications are specified by UML sequence diagrams with a set of boolean expressions. Based on analyzing the compatible integer state space of a real-time interface automaton network and its compatible reachability graph, we develop an algorithm to check the consistency between real-time component-based designs and the scenario-based timing specifications.

Keywords: embedded software designs, real-time systems, model checking, interface automata, UML sequence diagrams.

1 Introduction

For most embedded systems, the system correctness not only depends on functionality but also timeliness. And in recent years, a prevalent trend in the embedded computing area is that systems are more and more dominated by software. The software determines the functionality and quality of systems, hence presents lots of challenges to current software techniques. Using component-based design methodology in embedded software domain have been considered as an effective way for handling complexity, increasing system quality. One of the major goals in developing such systems is the validation that whether the component-based designs satisfy the specified timing specifications.

* Supported by the National Natural Science Foundation of China (No.60425204, No.60233020, and No.60273036), the National Grand Fundamental Research 973 Program of China (No.2002CB312001), and by Jiangsu Province Research Foundation (No.BK2004080).

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 395–404, 2005.
© IFIP International Federation for Information Processing 2005

In this paper, for real-time embedded software we consider the problem of checking the consistency of component-based designs for scenario-based timing specifications. Firstly the scenario-based timing specifications are specified by UML sequence diagrams[1] with a set of timming constraints. Then the interface automata[2] are used as a formal model to describe the dynamic behavior of software components, which is suitable for component-based verification. By introducing time intervals into interface automata, we make an extension for modelling real-time systems. And the component-based designs are modelled by real-time interface automaton networks, which includes a set of real-time interface automata synchronized by shared actions. Through analyzing the compatible integer state space of a real-time interface automaton network and its compatible reachability graph, finally we develop an algorithm for the timing behavior consistency checking problem.

2 UML Sequence Diagrams with Timing Constraints

In component-based system designs, the UML sequence diagrams can be used to describe the component interactions. It focus on the temporal order of the message flow. And sending a message, receiving a message, creating or deleting a component instance are all considered as events. According to [3], the semantics of each sequence diagram can be treated as a visual order sequence, which shows a partially ordered set of events displayed in the notation. Therefore, we can formalize the timing sequence diagrams and its behavior as follows:

Definition 1. A timing sequence diagram is a tuple $D = (C, E, M, F, W, S)$ where

- C is a finite set of components;
- E is a finite set of events;
- M is a finite set of messages. For any message $g \in M$, let $g!$ and $g?$ represent the sending and the receiving for g respectively. For any $e \in E$, it is corresponding to the sending or receiving for a message g, denoted by $\phi(e) = g!$ or $\phi(e) = g?$;
- $F : E \mapsto C$ is labelling function which maps each event $e \in E$ to a component $F(e) \in C$;
- W is a finite set whose elements are of the form (e, e') where e and e' are in E and $e \neq e'$, which represents a visual order displayed in D;
- S is a set of boolean expressions on event names, which represents the timing constraints enforced on D. □

Without loss of generality, let any timing constraint expression be of the form $c_0(t_{e_0} - t_{e_0'}) + c_1(t_{e_1} - t_{e_1'}) + \ldots + c_n(t_{e_n} - t_{e_n'}) \sim b$, where t_{e_0}, $t_{e_0'}$, t_{e_1}, $t_{e_1'}$, …, t_{e_n}, $t_{e_n'}$ present the time of the event occurrence respectively and c_0, c_1, …, c_n, b are real numbers ($b \neq \infty$), $\sim \in \{\leq, <\}$.

Definition 2. For any $D = (C, E, M, F, W, S)$, an event sequence $e_0 \hat{} e_1 \hat{} \ldots \hat{} e_m$ is a *trace* of D iff the following conditions hold:

– all events in E occur in the sequence, and each event occurs only once, i.e. $\{e_0, e_1, \ldots, e_m\} = E$ and $e_i \neq e_j$ for any i, j $(i \neq j,\ 0 \leq i, j \leq m)$; and
– $e_0, e_1, \ldots, e_m$ satisfy the visual order defined by W, i.e. for any e_i and e_j, if $(e_i, e_j) \in W$, then $0 \leq i < j \leq m$. □

We use *timed event sequences*, with the form of $(e_0, \tau_0)\hat{}(e_1, \tau_1)\hat{} \ldots \hat{}(e_m, \tau_m)$, to represent the behavior of real-time systems, in which each event e_i occurs with a relative time stamp τ_i. It means that e_0 takes place τ_0 time units after the system starts, then e_1 takes place τ_1 time units after e_0 takes place, so on and so forth, at last e_m takes place τ_m time units after e_{m-1} takes place.

Definition 3. A timed event sequence $\sigma = (e_0, \tau_0)\hat{}(e_1, \tau_1)\hat{} \ldots \hat{}(e_m, \tau_m)$ is a behavior of a timing sequence diagram $D = (C, E, M, F, W, S)$ iff $e_0\hat{}e_1\hat{} \ldots \hat{}e_m$ is a trace of D, and $\tau_0, \tau_1, \ldots, \tau_m$ satisfy the timing constraints described by S, i.e. for any boolean expression $\sum_{i=0}^{n} c_i(t_{f_i} - t_{f_i'}) \sim b$ in S, there is $\sum_{i=0}^{n} c_i \Delta_i \sim b$ where for each i $(0 \leq i \leq n)$, if $f_i = e_j$ and $f_i' = e_k$, then $\Delta_i = \sum_{u=k+1}^{j} \tau_u$ $(j > k)$or $\Delta_i = -\sum_{u=j+1}^{k} \tau_u$ $(j < k)$. □

3 Real-Time Interface Automata and Component-Based Designs

3.1 Real-Time Interface Automata

Interface automaton[2] is a light-weight formal language to describe the temporal aspects of software component interfaces. Specifically, it's designed to capture effectively both input assumptions and output guarantees about the order of the interactions between component and its environment. To further describe real-time properties, we need to extend the interface automata with timeliness. In the following, a *real-time interface automaton (RIA)* model is introduced, in which for any transition there is a time interval constraint.

Definition 4. A *RIA* is a tuple $P = (V_P, v_P^{Init}, A_P, I_P, \Gamma_P)$ where:

– V_P is a finite set of states, each state $v \in V_P$;
– $v_P^{Init} \in V_P$ is the initial state;
– A_P is the set of all actions, which includes A_P^I, A_P^O and A_P^H, the mutually disjoint set of input, output and internal actions respectively.
– I_P is a finite set of time intervals; each interval has the form of $[x, y]$ where x, y are nonnegative integer numbers $(x \leq y$, y may be $\infty)$;
– $\Gamma_P \subseteq V_P \times A_P \times I_P \times V_P$ is a set of transitions. □

An action $a \in A_P$ is *enabled* at state $v \in V_P$ if there is a transition $(v, a, [x, y], v') \in \Gamma_P$. An interface automaton is *non-input-enabled*; that is, it's not require that $A_P^I(v) = A_P^I$ for all states $v \in V_P$.

We use a *timed state sequence* of the form $v_0 \xrightarrow{a_0,\tau_0} v_1 \xrightarrow{a_1,\tau_1} \ldots \xrightarrow{a_{m-1},\tau_{m-1}} v_m \xrightarrow{a_m,\tau_m} v_{m+1}$ to describe a behavior of a *RIA*, which means that the system

is starting at state v_0, after τ_i time units changing to v_1 through the transition ignited by the action a_0 and staying there for τ_1 time units, and so on. As the same we have mentioned before, each $\tau_i (0 \leq m)$ is a nonnegative integer number representing the relative time stamp.

Definition 5. For a real-time interface automaton $P = (V_P, v_P^{Init}, A_P, I_P, \Gamma_P)$, a timed state sequence $v_0 \xrightarrow{a_0,\tau_0} v_1 \xrightarrow{a_1,\tau_1} \ldots \xrightarrow{a_{n-1},\tau_{n-1}} v_n \xrightarrow{a_n,\tau_n} v_{n+1}$ is a behavior of P iff $v_0 = v_P^{Init}$, and for each $i (0 \leq i \leq n)$, there is $(v_i, a_i, [x_i, y_i], v_{i+1}) \in \Gamma_P$, $x_i \leq \tau_i \leq y_i$. □

3.2 Real-Time Interface Automaton Networks

We use *real-time interface automaton networks (RIAN)* to model the component-based designs for a real-time embedded software. It consists of a set of real-time interface automaton which represent the dynamic behaviors of software components. Since an input action of one interface automaton may concide with an output action of the other one, these two interface automata will synchronize on such shared actions, asynchronously interleaving on other actions. Those synchronized actions between any two interface automata (P_i, P_j) are denoted by $shared(P_i, P_j) = A_{P_i} \cap A_{P_j} = (A_{P_i}^O \cap A_{P_j}^I) \cup (A_{P_i}^I \cap A_{P_j}^O)$. More details about the composition of interface automata can be refereed to [2].

For solving the verification problem, we give the formalism of a *RIAN* as follows, including the states , actions and transitions.

Definition 6. Let $N = (K, Z)$ be a *RIAN* where $K = \{P_1, P_2, \ldots, P_n\}$, each $P_i = (V_{P_i}, v_{P_i}^{Init}, A_{P_i}, I_{P_i}, \Gamma_{P_i})$, and $Z = \{shared(P_i, P_j) \mid 1 \leq i, j \leq n, i \neq j\}$ is a set of all shared actions. The states and actions are defined as below:

- an *untimed state* $\overline{v}$ of N is in $V_{P_1} \times V_{P_2} \times \ldots \times V_{P_n}$, that is, $\overline{v} = (v_1, v_2, \ldots, v_n)(v_i \in V_{P_i}, 1 \leq i \leq n)$. The *initial untimed state* of N is $\overline{v}_N^{Init} = (v_{P_1}^{Init}, v_{P_2}^{Init}, \ldots, v_{P_n}^{Init})$;
- a *state* u of N is a pair $u = (\overline{v}, c)$ where $\overline{v} = (v_1, v_2, \ldots, v_n)$ is an untimed state of N and $c : \{v_i | 1 \leq i \leq n\} \mapsto \{R^+\} \cup \{0\}$ is called the *clock function* which maps each v_i to a nonnegative real number that indicates for the P_i how long the system has been staying at v_i. The *initial state* of N is $u_N^{Init} = (\overline{v}_N^{Init}, c^{Init})$ where $\overline{v}_N^{Init}$ is the initial untimed state of N and $c^{Init}(v_{P_i}^{Init}) = 0$ for any $i (1 \leq i \leq n)$. The set of states of N is denoted by U_N;
- the set of actions of N is $A_N = A_N^I \cup A_N^O \cup A_N^H$, where the set of input actions is $A_N^I = \left(\bigcup_{1 \leq i \leq n} A_{P_i}^I\right) / Z$, the set of output actions is $A_N^O = \left(\bigcup_{1 \leq i \leq n} A_{P_i}^O\right) / Z$, and the set of internal actions is $A_N^H = \left(\bigcup_{1 \leq i \leq n} A_{P_i}^H\right) \cup Z$.
- the set of time intervals of N is $I_N = \bigcup_{1 \leq i \leq n} I_{P_i}$. □

Definition 7. Let $N = (K, Z)$ be a *RIAN* where $K = \{P_1, P_2, \ldots, P_n\}$ and $P_i = (V_{P_i}, v_{P_i}^{Init}, A_{P_i}, I_{P_i}, \Gamma_{P_i})(1 \leq i \leq n)$, and $u = (\overline{v}, c)$ and $u' = (\overline{v}', c')$ be its states where $\overline{v} = (v_1, v_2, \ldots, v_n)$ and $\overline{v}' = (v_1', v_2', \ldots, v_n')$. Then the system can change from the state u to u' by an transition within a delay d(denoted by $u \xrightarrow{a,d} u'$), if one of the following conditions holds:

- for an action $a \notin Z$, there is a transition $(v_k, a, [x_k, y_k], v'_k) \in \Gamma_{P_k}$ $(1 \leq k \leq n)$ satisfying $x_k \leq c(v_k) + d \leq y_k$ and $c'(v'_k) = 0$. At the same time $v_i = v'_i$ and $c'(v_i) = c(v_i) + d$ for any i $(i \neq k, 1 \leq i \leq n)$; or
- for an action $a \in \mathit{shared}(P_i, P_j)(1 \leq i, j \leq n, i \neq j)$, there are transitions $(v_i, a!, [x_i, y_i], v'_i) \in \Gamma_{P_i}$ ($a!$ denote a is an output action) and $(v_j, a?, [x_j, y_j], v'_j) \in \Gamma_{P_j}$ ($a?$ denote a is an input action) satisfying $x_i \leq c(v_i) + d \leq y_i$, $x_j \leq c(v_j) + d \leq y_j$, $c'(v'_i) = 0$ and $c'(v'_j) = 0$. At the same time, $v_k = v'_k$ and $c'(v_k) = c(v_k) + d$ for any $k(k \neq i, j, 1 \leq k \leq n)$.

Then a behavior of N is a timed state sequence: $u_0 \overset{a_0,d_0}{\longrightarrow} u_1 \overset{a_1,d_1}{\longrightarrow} \ldots \overset{a_{n-1},d_{n-1}}{\longrightarrow} u_n \overset{a_n,d_n}{\longrightarrow} u_{n+1}$ where u_0 is the initial state of N. □

4 Checking Real-Time Component-Based Designs for Scenario-Based Specifications

Now, for a real-time embedded software the component-based designs are modelled by a real-time interface automaton network, and the scenario-based timing specifications are specified by the sequence diagrams with timing constraints. The problem we concern is to check if the real-time interface automata interact according to the scernio specified by the timing sequence diagrams.

Firstly, we need to give the relation between a behavior of the *RIAN* and a trace of the timing sequence diagram. Let $N = (K, Z)$ be a *RIAN*, and $\varrho = u_0 \overset{a_0,d_0}{\longrightarrow} u_1 \overset{a_1,d_1}{\longrightarrow} \ldots \overset{a_{n-1},d_{n-1}}{\longrightarrow} u_n \overset{a_n,d_n}{\longrightarrow} u_{n+1}$ be a behavior of N. Then the corresponding timed action sequence of ϱ is $(a_0, d_0)^\frown(a_1, d_1)^\frown \ldots ^\frown(a_n, d_n)$, and those timed internal action (a_i, d_i) (which satisfies $a_i \in Z$) can be replaced by a pair of timed input and output actions $(a_i!, d_i)^\frown(a_i?, 0)$, as a result we can get an timed action sequence which only contains input and output actions. In fact, each input(resp. output) action a can be considered as an input(resp. output) event e, thus a corresponding timed event sequence, with a form of $(e_0, \tau_0)^\frown(e_1, \tau_1)^\frown \ldots ^\frown(e_r, \tau_r)(r \geq n)$, is obtained which is called the *trail* of ϱ. On the other hand, notice that any message appeared in a sequence diagram actually should be one of the shared actions in N, and each event such as sending message or receiving message should be one of the output or input actions of the corresponding component's *RIA*. However, in the formal definition of the sequence diagram $D = (C, E', M, F, W, S)$ an event $e' \in E'$ just represents an event name and the event content is denoted by $\phi(e')$. Thus if we want to compare the events between the trail of ϱ and the trace of D, it need to compare e with $\phi(e')$.

Let ϱ be a behavior of N, σ be the trail of ϱ, and σ_1 be a subsequence of σ of the form $(e_0, \tau_0)^\frown(e_1, \tau_1)^\frown \ldots ^\frown(e_m, \tau_m)$. For a trace of D with the form of $f_0{}^\frown f_1{}^\frown \ldots ^\frown f_n$, if there are $e_{k_0}, e_{k_1}, \ldots, e_{k_n}$ satisfying that (1) $0 = k_0 < k_1 < \ldots < k_n = m$, (2)$\phi(f_i) = e_{k_i}$ for any i $(0 \leq i \leq n)$, and (3)for any i $(0 \leq i \leq n)$, for any $j \neq k_p$ $(0 \leq j \leq m, 0 \leq p \leq n)$, $\phi(f'_i) \neq e_j$, then the subsequence σ_1 is considered as a *projection* of σ over D. In fact, σ_1 shows that an scenario described by D *occurs* exactly in behavior ϱ of N. Based on above discussion,

we give the formal description of the timing consistency verification problem as below:

Definition 8. Let N be a *RIAN*, and $D = (C, E, M, F, W, S)$ be a timing sequence diagram. N *satisfies* D iff the following conditions hold:

- the scenario described by D occurs in a behavior of N; and
- for any behavior of N, the projection of its trail over D, which have a form of $(e_k, \tau_k)\hat{}(e_{k+1}, \tau_{k+1})\hat{} \ldots \hat{}(e_r, \tau_r)$, satisfies the timing constraints of D, i.e. for any boolean expression $\sum_{i=0}^{n} c_i(t_{f_i} - t_{f_{i'}}) \sim b$ in S, there is $\sum_{i=0}^{n} c_i \Delta_i \sim b$, where for each i $(0 \leq i \leq n)$ if $\phi(f_i) = e_g$ and $\phi(f_i') = e_h (k \leq g, h \leq r)$, then $\Delta_i = \sum_{u=h+1}^{g} \tau_u$ $(g > h)$or $\Delta_i = -\sum_{u=g+1}^{h} \tau_u$ $(g < h)$. □

It's clearly that the state space of *RIAN* is infinite, so the key point is to find a way for verification in a finite manner.

4.1 Constructing an Integer Reachability Graph

We introduce the *integer behavior* as follows. Let $\varrho = u_0 \xrightarrow{a_0,d_0} u_1 \xrightarrow{a_1,d_1} \ldots \xrightarrow{a_{n-1},d_{n-1}} u_n \xrightarrow{a_n,d_n} u_{n+1}$ is a behavior of a *RIAN*. If d_i is an integer for any i $(0 \leq i \leq n)$, the ϱ is called an *integer behavior* . It follows that any state $((v_1, v_2, \ldots, v_m), c)$ occurring in an integer behavior satisfies $c(v_j)$ is an integer for any j $(1 \leq j \leq m)$, which is called *integer state*.

Theorem 1. Let N be a *RIAN*, and $D = (C, E, M, F, W, S)$ be a timing sequence diagram. N satisfies D iff the scenario described by D occurs in an integer behavior of N, and for any integer behavior of N, the projection of its trail over D satisfies the timing constraints of D. □

Then, for above checking problem, we only need to consider the integer behaviors of the *RIAN*.

To deal with the infinite case in some time intervals, we represent a *time bound* of N by $K = \max\{X_{max}, Y_{max}\}$, where $X_{max} = \max\{x_i | [x_i, \infty) \in I_N (0 \leq i \leq |I_N|)\}$ and $Y_{max} = \max\{y_i | [x_i, y_i] \in I_N (x_i \leq y_i, y_i \neq \infty), 0 \leq i \leq |I_N|\}$. Then let $u = (\overline{v}, c)$ and $u' = (\overline{v}', c')$ be the integer states of N where $\overline{v} = (v_1, v_2, \ldots, v_n)$ and $\overline{v}' = (v_1', v_2', \ldots, v_n')$. and define a time-zone relation R_t over U_N as below:

$$R_t = \left\{ < u, u' > \left| \begin{array}{l} \overline{v} = \overline{v}' \text{ and for each } v_i (1 \leq i \leq n), \text{ either} \\ c(v_i) = c'(v_i) \text{ or } c(v_i) > K \wedge c'(v_i) > K \end{array} \right. \right\}.$$

R_t is an equivalent relation over U_N, and the number of the equivalent class is finite. We use $[u]$ to represent the equivalent class, that is,$[u] = \{u' | u R_t u'\}$, and denote by $[N]$ the finite space which comprise of the equivalent classes under R_t.

Theorem 2. Let N be a *RIAN*, and u_1 and u_2 are the integer states such that $u_1 R_t u_2$. Then there is an integer state u_1' such that $u_1 \xrightarrow{a,d} u_1'$ iff there is an integer state u_2' such that $u_2 \xrightarrow{a,d} u_2'$ and $u_1' R_t u_2'$. □

It shows that the set of integer behaviors of $[N]$ are consistent with those of N.

Based on the time-zone equivalent relation, we can construct a reachability graph $G = \{V_G, L_G\}$ for a *RIAN* N as follows, where V_G is a set of nodes and L_G is a set of edges:

1. for the initial state u_N^{Init} of N, $[u_N^{Init}]$ is in the set V_G, which is called the *initial node*;
2. let $[u]$ be in the set V_G, and K be the time bound of N. For an event a, for an integer d $(0 \leq d \leq K+1)$, if $u \xrightarrow{a,d} u'$ then $[u']$ is in V_G, and the edge $[u] \xrightarrow{a,d} [u']$ is in the set L_G. When $a \in A_N^I$ (resp. A_N^O, A_N^H), the edge $[u] \xrightarrow{a,d} [u']$ is called input (resp. output, internal) edge. We use L_G^I,L_G^O and L_G^H to represent the set of input, output and internal edges respectively.

Let $\rho = l_0 \hat{} l_1 \hat{} \ldots \hat{} l_n (l_i : [u_i] \xrightarrow{a,d} [u_{i+1}] \in L_G, 0 \leq i \leq n)$ be a path of G which is an edge sequence starting from the initial node. From its corresponding timed action sequence $(a_0, d_0) \hat{} (a_1, d_1) \hat{} \ldots \hat{} (a_{n-1}, d_{n-1})$, we can replace those (a_i, d_i) satisfying that a_i is a shared action with a pair of input and output actions $(a_i!, d_i) \hat{} (a_i?, 0)$, as the same way mentioned before. Then we can also get a timed event sequence, with a form of $(e_0, \tau_0) \hat{} (e_1, \tau_1) \hat{} \ldots \hat{} (e_s, \tau_s)(s \geq n-1)$, which is called the *trail* of path ρ. Any edge sequence $l_i \hat{} l_{i+1} \hat{} \ldots \hat{} l_{i+k}$ $(0 \leq i \leq n-1-k)$is called a *subpath* of ρ. We denote the trail of a subpath $l_i \hat{} l_{i+1} \hat{} \ldots \hat{} l_j$ by $\sigma(l_i, l_j)(i \leq j)$, the tail of path ρ by σ_ρ or $\sigma(l_0, l_n)$. Thus each label on the edges either contains an input(output) event or a pair of input and output events. It denotes by $\psi(l_k)$ the event pair corresponding to l_k.

4.2 Eliminating the Illegal States

Since the interface automaton is *non-input-enabled*, there will be some *illegal states* during the composition of two interface automata. The illegal state represents that one automaton in *RIAN* may produce an output event that is an input event of another automaton, but is not accepted by the latter one on that state. The formal definition is given below:

Definition 9. Let $N = (K, Z)$ be a *RIAN* where $K = \{P_1, P_2, \ldots, P_n\}$ and $P_i = (V_{P_i}, v_{P_i}^{Init}, A_{P_i}, I_{P_i}, \Gamma_{P_i})(1 \leq i \leq n)$,the set of illegal states is shown as:

$$illegal(N) = \left\{ ((v_1, v_2, ., v_n), c) \in N \;\middle|\; \begin{array}{l} \exists (v_i, v_j)(i \neq j, 1 \leq i, j \leq n), \\ \exists a \in shared(P_i, P_j), \\ \left(\begin{array}{c} a \in A_{P_i}^O(v_i) \wedge a \notin A_{P_j}^I(v_j) \\ \vee \\ a \in A_{P_j}^O(v_j) \wedge a \notin A_{P_i}^I(v_i) \end{array} \right) \end{array} \right\} \square$$

Based on the theorem 2 and the above definition, the following claim can be established: if u is an integer state satisfying $u \in illegal(N)$, then all the integer states in the time-zone equivalent class $[u]$ are illegal. In this case, $[u]$ is called an *illegal time-zone equivalent class*. And we use $illegal(G)$ to represent those nodes of reachability graph G which corresponds to the set of illegal time-zone equivalent classes of $[N]$.

Based on the reachability graph G, using the same algorithm framework in [2], we can get a set of compatible time-zone equivalent classes of $[N]$ (denoted by $\Pi_{[N]}$). The basic idea of the algorithm is described briefly as follows. Firstly, an operator $OH_{pre} : 2^{V_G} \mapsto 2^{V_G}$ is defined as: for all sets $\Lambda \subseteq V_G$, $OH_{pre}(\Lambda) = \{\mu \in V_G | \exists(\mu \xrightarrow{a,d} \nu) \in L_G^O \cup L_G^H; \nu \in \Lambda\}$. Then the maximum fixpoint of operator OH_{pre} can be computed over reachability graph G by assigning Λ an initial value *illegal (G)*.

If the set $\Pi_{[N]}$ is empty, from the optimistic view of interface automata, it indicates that there are no helpful environments existing which can prevent the *RIAN* from entering those illegal states in the running time. In this case, the result of the verification problem is false. If the $\Pi_{[N]}$ is nonempty, it implies that there are some kinds of helpful environments making the *RIAN* to work normally; then we only need to concern about the behaviors of the compatible time-zone equivalent class space, which is denoted by $com([N])$. That is, for the verification problem, checking the integer behaviors in $com([N])$ is enough.

Theorem 3. Let N be a *RIAN*. N satisfies a timing sequence diagram D iff:

- there is an integer behavior in the $com([N])$, which contains an occurrence of the scenario described by D; and
- for any integer behavior of $com([N])$, the projection of its trail over D satisfies the timing constraints in D. □

Corresponding to $com([N])$, we use the $com(G)$ to represent the *compatible reachability graph*, which only contains the nodes comprised of $\Pi_{[N]}$. Obviously any path in $com(G)$ represents an integer behavior of $com([N])$. However, during the construction of G there is a constraint enforced on $d(0 \leq d \leq K+1)$, and it leads to a result that an integer behavior in $com([N])$ may not correspond to a path in $com(G)$. So, for any path ρ of $com(G)$ and its tail σ_ρ with a form of $(e_0, \tau_0)^\wedge(e_1, \tau_1)^\wedge \ldots ^\wedge (e_m, \tau_m)$, we construct a set of $\theta(\sigma_\rho)$ as below:

$$\theta(\sigma_\rho) = \left\{ (e_0, \tau_0')^\wedge(e_1, \tau_1')^\wedge \ldots ^\wedge(e_m, \tau_m') \;\middle|\; \begin{array}{l} \text{for any } i(0 \leq i \leq m), \text{ if } \tau_i = K+1, \\ \text{then } \tau_i' \in [K+1, \infty), \text{ else } \tau_i' = \tau_i. \end{array} \right\}.$$

Now it is clear that every timed event sequence in $\theta(\sigma_\rho)$ is the tail of an integer behavior of $com([N])$, and for any timed event sequence σ' which is the trail of an integer behavior of $com([N])$, there is a path ρ' in $com(G)$ such that $\sigma' \in \theta(\sigma_{\rho'})$. So we can solve the problem of checking a *RIAN* for a timing sequence diagram D by checking every path in $com(G)$.

4.3 Checking the Consistency

Even in the compatible integer state space, the number of paths in $com(G)$ may be infinite and the length of a path in $com(G)$ may be also infinite, so we have to solve the problem basing on a finite set of finite paths in $com(G)$. For that purpose, a *projection path* is introduced as follows. For a sequence diagram $D = (C, E, M, F, W, S)$, a path $\rho = l_0{}^\wedge l_1{}^\wedge \ldots {}^\wedge l_n$ with a trail $\sigma(l_0, l_n)$ is called a

```
currentpath := ⟨[u0]⟩; loopset := ∅;
repeat
  node := the last node of currentpath;
  if node has no new successive node then delete the last node of currentpath
  else begin node := a new successive node of node;
          if node is such that there is a simple flat loop for D in the path
            of currentpath^node
          then put the loop into loopset
          else append node to currentpath;
       end
until currentpath = ⟨⟩;
```

```
currentpath := ⟨[u0]⟩; is_projection_path:=false;
repeat
  node := the last node of currentpath;
  if node has no new successive node then delete the last node of currentpath
  else begin node := a new successive node of node;
          if node satisfies that the path currentpath^node is a projection path
          then begin check if the projection path satisfies D;
                     if no, return false;
                     is_projection_path :=true;
                     check if the projection path is a flaw projection path for D;
                     if yes, return false;
               end
          if node satisfies that currentpath^node is a prefix of projection path
          then append node to currentpath;
       end
until currentpath = ⟨⟩;
if is_projection_path then return true else return false.
```

Fig. 1. Algorithm to check a *RIAN* for a timing sequence diagram D

projection path if it satisfies the following conditions: (1) the trail $\sigma(l_i, l_n)(0 \leq i \leq n)$ is a projection of $\sigma(l_0, l_n)$ over D,(2) for any $j, k(0 < j < k < i)$, $l_j \neq l_k$; and (3)for any neighbor $j, k(i \leq j < k \leq n, \psi(l_j) \in E, \psi(l_k) \in E)$, $l_g \neq l_h$ $(j < g < h < k)$. If the timed event sequence $\sigma(l_i, l_n)$ satisfies the boolean timing expressions in D, we say that the projection path *satisfies* D. It is clearly that the length of a projection path in $com(G)$ is finite and the number of projection paths in $com(G)$ is also finite.

And we further consider the loops in the $com(G)$. For a subpath $\rho_1 = l_j \hat{} l_{j+1} \hat{} \ldots \hat{} l_k$, where $l_i : [u_i] \xrightarrow{a_i, d_i} [u_{i+1}](j \leq i \leq k)$. If $l_j = l_k$ then we say that ρ_1 is a *loop*, and $d_{j+1} + d_{j+2} + \ldots + d_k$ is the *elapsed time* on ρ_1, denoted by $T(\rho_1)$. If for any $p, q(j \leq p < q < k)$, there is $l_p \neq l_q$, then we say that ρ_1 is a *simple loop*. For a $D = (C, E, M, F, W, S)$, if any $e \in E$ does not occur in the trail $\sigma(l_j, l_k)$ of ρ_1, we say that ρ_1 is a *flat loop* for D.

Let $\rho = l_0 \hat{} l_1 \hat{} \ldots \hat{} l_m$ be a projection path in $com(G)$, if there is a flat loop $\rho'(T(\rho') > 0)$ for D which contains the edge $l_i : [u_i] \xrightarrow{a_i, d_i} [u_{i+1}]$ $(0 \leq i \leq m)$ and d_i occurs in $\Delta_p (0 \leq p \leq n, c_p \Delta_p \geq 0)$ (see Definition 8), then the path ρ is considered as a *flaw projection path* for D. The edge l_i is called a *flaw point* in ρ. Notice that from a flaw projection path, we can always construct a path in $com(G)$ which does not satisfy the timing constraints in D.

Theorem 4. Let N be a *RIAN*, and G be its integer reachability graph. For a timing sequence diagram D, N satisfies D iff:

- there is a projection path in $com(G)$;
- any projection path in $com(G)$ satisfies D; and
- there is not any flaw projection path for D in $com(G)$. □

Based on the above theorem, we can develop an algorithm for the verification problem, which is depicted in Fig.1. The algorithm consists of two phases which are implemented by depth first search. The first search is to get all simple flat loops for D which are used for checking if there is a flaw point in a projection path. The second one is to find out all the projection paths in $com(G)$ and to check them for D. Since there is only a finite number of projection paths in $com(G)$ and the algorithm is based on depth first search method, the complexity of the algorithm is proportional to number and length of projection paths in $com(G)$.

5 Conclusion

Based on the solution in this paper, we are developing a prototype tool. For future work, we will do more case studies in practical use, and consider the resource and energy analysis in the embedded real-time software systems.

References

1. G. Booch, J. Rumbaugh and I. Jacobson. *The Unified Modeling Language User Guide*, Addison-Wesley, 1999.
2. L. de Alfaro and T. A. Henzinger. Interface Automata. In *Proc. of ESEC/FSE 01*, Austria, 2001
3. Doron A. Peled. *Software Reliability Methods*. pp300-305, Springer, 2001.
4. J. Seemann, J. WvG. Extension of UML Sequence Diagrams for Real-Time Systems. In *Proc. International UML Workshop*, LNCS 1618, pp240-252, 1998.
5. Rajeev Alur and Mihalis Yannakakis. Model Checking of Message Sequence Charts. In *Proc. the 10th International Conference on Concurrency Theory*, LNCS 1664, pp114-129, 1999.
6. Thomas Firley, Michaela Huhn, Karsten Diethers, et al. Timed Sequence Diagrams and Tool-Based Analysis - A Case Study. In *Proc. of the Second International Conference on UML (UML99)*, LNCS 1732, pp.645-660, 1999.
7. L. de Alfaro, T. A. Henzinger, and M. Stoelinga. Timed Interfaces. In *Proc. of EMSOFT 2002*, LNCS 2491,pp108-122, 2002.
8. Alexandre David, M. Oliver, and Wang Yi. Formal Verification of UML Statecharts with Real-Time Extensions. In: *FASE2002*, LNCS 2306, 2002, pp.218-232.
9. Hu Jun, Yu Xiaofeng, Zhang Yan, et al. Scenario-Based Verification for Component-Based Embedded Software Designs. In *Proc. of ICPP 2005 Workshops*,pp.240-247,June 2005, IEEE Computer Society Press.

Dependable Polygon-Processing Algorithms for Safety-Critical Embedded Systems

Jens Brandt and Klaus Schneider

University of Kaiserslautern,
Reactive Systems Group, Department of Computer Science,
P.O. Box 3049, 67653 Kaiserslautern, Germany
http://rsg.informatik.uni-kl.de

Abstract. Algorithms that process geometric objects become more and more important for many safety-critical embedded systems, e.g. for motion planning or collision detection, where correctness is indispensable. The main challenge to demonstrating correctness is the consistent handling of degenerate cases like collinear line segments. In this paper, we therefore propose the use of an interactive theorem prover to develop dependable geometry algorithms for safety-critical embedded systems. Our solution is based on the use of a three-valued logic to make degenerate cases explicit. Using the theorem prover, we are not only able to prove the correctness of the obtained algorithms, but also to directly derive a software library of provably correct geometry algorithms for safety-critical applications.

1 Introduction

Applications like motion planning in robotics or collision detection of autonomous vehicles have to observe and to control the positions of certain objects in a particular environment. To this end, it is often sufficient to model the environment as a two-dimensional plane and the objects can be approximated as polygons on that plane. For this reason, fundamental algorithms of computational geometry [7] like polygon clipping or computing convex hulls can be used to develop embedded systems for these applications. As these embedded systems work in safety-critical domains, where even human live could be endangered, it is mandatory to guarantee their correctness. For this reason, it is necessary to built up specification languages and verification formalisms that allow the designers of such systems to reason about their correctness.

Geometric problems appear to be easy, since they can be visualised in a natural and intuitive way. However, at a second glance, even simple definitions turn out to be more complicated than expected: For example, what is the intersection point of two lines, if both lines are identical? For this reason, most algorithms work under certain preconditions like ‘all points are pairwise distinct’ or ‘no three points are collinear’.

In order to guarantee that all possible cases are consistently handled and that the used algorithms work as expected, we recommend the use of interactive theorem provers to reason about required definitions and algorithms. However, establishing consistent definitions for geometric primitives that can be universally used for all algorithms and even degenerate inputs turns out to be a great challenge [8]. For example,

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 405–417, 2005.
© IFIP International Federation for Information Processing 2005

to define whether a point on the edge of a polygon belongs to the interior or not, yields essential problems when this definition is used in one or the other algorithm. These problems impose inherent difficulties to develop a software library for safety-critical embedded systems to process geometric data.

In our opinion, the best solution is to make degenerate cases explicit, which requires the introduction of a third truth value for geometric primitives that normally are either true or false. We therefore extended the interactive theorem prover HOL [10] by a theory on two-dimensional analytic geometry [3] that is based on a three-valued logic. We formalised geometric objects, geometric primitives, and finally entire algorithms, so that we are able to reason about the correctness of geometric computations. As a first experiment, we successfully verified the Cohen-Sutherland clipping algorithm [9].

Though reasoning about geometric problems has a long tradition, our work deviates from existing work in several ways. In particular, Wu's work [21, 6] on translating geometric propositions to algebraic forms, i.e. equations between polynomials is well-known. However, this approach is limited to the automated solution of particular instances of geometric problems, while we have to argue about the problems (and their algorithmic solutions) themselves. The work closest to ours is [15], where also an interactive theorem prover is used to formally reason about geometric problems. However, in contrast to our work, they used two-valued logic to formalise geometric primitives. To circumvent problems of degenerate cases, they modified definitions or used techniques like perturbation which, however, turned out to be problematic in practise [5].

In this paper, we therefore propose to use three-valued logic for the implementation of algorithms that process geometric data. Three-valued logic has proved to be an adequate tool in many areas of computer science [4, 20, 13, 2, 18, 16, 1, 19]. We show that three-valued logic is well suited to define geometric primitives so that degenerate cases are consistently and concisely handled. To guarantee the correctness of the derived algorithms, we employ the interactive theorem prover HOL [10] to set up a library on two-dimensional, linear objects like lines, segments, and polygons [3]. Based on this library for the theorem prover, we derive a dependable software library that provides algorithms for geometric problems that have to be solved in upcoming embedded systems.

The paper is organised as follows: In Section 2, the specification and verification of polygon processing algorithms is described. Section 3 illustrates our formalisation approach by the verification of the Cohen-Sutherland algorithm for line clipping. Section 4 sketches the implementation of a software library for algorithms of computational geometry based on our three-valued formalisations. Finally, Section 5 draws some preliminary conclusions.

An extended version of this paper, the proof code, additional material, as well as detailed descriptions of all theories and examples described in this paper are available on our website.

2 Specification

In this section, we describe our three-valued formalisation of analytic geometry. We first consider degenerate cases and three-valued logic (Section 2.1) and then more complex geometric objects and primitives (Section 2.2).

2.1 Degenerate Cases and Three-Valued Logic

Most algorithms of computational geometry are designed for the 'general case', thus excluding so-called degenerate cases: Depending on the algorithm, several preconditions are assumed, e.g. that no points coincide, that given lines are not parallel, or that no three lines intersect in a common point [8, 14]. Handling degenerate cases is a well-known problem in computational geometry.

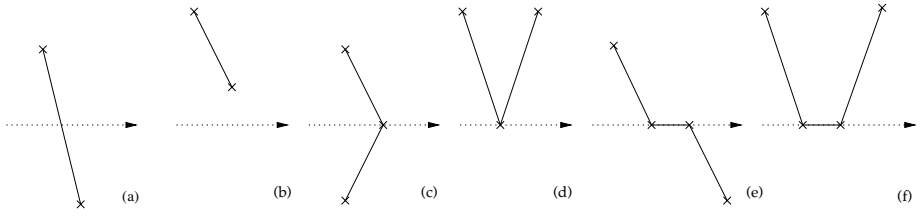


Fig. 1. Winding number algorithm

As an example, consider the winding number algorithm: It calculates the winding number of a point p relative to a polygon P, i.e. how often the directed sequence of connected line segments of P turns around p (see [11] for a precise definition). To this end, it counts the intersections of an arbitrary ray starting in p with edges of the polygon P. If the intersected edge runs from the bottom to the top, it is counted positively - in the opposite direction negatively. Figure 1 shows the possible cases, where the ray is drawn with a dotted line and some edges of polygon P are drawn with straight lines: (a) and (b) show simple cases without problems, while (c) to (f) show degenerate cases, i.e. a vertex or an edge of the polygon is on the ray. Depending on the position of the adjoining edges of the polygon, the situation must be counted as an intersection or not. In the presented examples, cases (c) and (e) have to be defined as intersections, whereas cases (d) and (f) should not be intersections.

The explicit treatment of degeneracies requires substantial effort. In particular, to decide whether all cases have been considered is often unclear and nontrivial. Simple solutions like perturbation or other techniques have shown not to be as applicable as desired [5], so that degeneracies still form an impediment for the development of dependable algorithms: On the one hand, the enormous number of different degenerate cases does not allow an explict treatment, and on the other hand, consistent definitions of geometric primitives that implicitly handle all degenerate cases for all algorithms in an adequate way are not available, and probably do not exist: For example, consider the problem to determine whether a point is inside or outside a polygon. Assume that the points on the edge are considered to be outside the polygon (i.e. polygons are 'open' point sets). If we calculate the difference of two polygons as a set difference, the result is possibly a polygon that contains points on its edge.

For this reason, we propose the use of three-valued logic to handle degeneracies. Three-valued logic has already proven to be useful in many areas, e.g. for the analysis of asynchronous circuits [13, 4, 20], for program analysis [16, 2, 17, 18], for the analysis

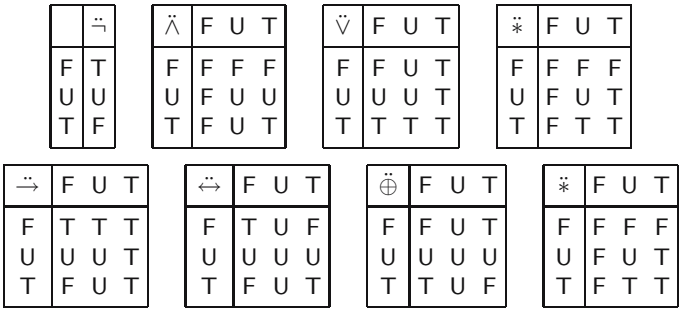


Fig. 2. Truth tables of three-valued operators

of cache behaviour [1], and temporal properties of programs [19]. We show that three-valued logic is also a natural foundation to specify geometric problems. Morevover, we do not only use three-valued logic for the analysis of (two-valued) algorithms, but instead propose to implement all geometric primitives directly by three-valued logic. This allows us to describe algorithms in a concise way without having the need to enumerate many tedious degenerate cases. Clearly, these cases do not disappear, but three-valued logic allows us to implicitly treat them in a systematic way. An intuitive view on the third truth value is obtained by the analogy to exception handling in some programming languages: At the point of time where an error occurs, it is not clear how to handle it. Thus, an exception is thrown, which is finally caught by a function that has the necessary context knowledge to handle the problem. Analogously, the degenerate case is passed through all functions until the knowledge of the context is sufficient to resolve the problem.

Reconsider the 'point-in-polygon problem': The area of a polygon is described by a function that maps each point of the plane to one of the three truth values: *true* (T) is assigned to all points inside, *false* (F) to all points outside, and *degenerate* (U) is assigned to all points on the edge of the polygon. These considerations give rise to the definitions of the basic three-valued connectives $\ddot{\neg}$, $\ddot{\wedge}$ and $\ddot{\vee}$ shown in Figure 2, which have already been used by Kleene [12]. We introduce further operators (see Figure 2) like implication $\ddot{\rightarrow}$, equivalence $\ddot{\leftrightarrow}$, exclusive-or $\ddot{\oplus}$ and a modified conjunction $\ddot{*}$ (the meaning of which will be explained in Section 2.2).
Moreover, we extend the theory by existential and universal quantification:

$$\texttt{exists3} \vdash_{\text{def}} \ddot{\exists} P = \mathbf{if}\ (\exists x.P(x) = \mathsf{T})\ \mathbf{then}\ \mathsf{T}\ \mathbf{else}\ (\mathbf{if}\ (\forall x.P(x) = \mathsf{F})\ \mathbf{then}\ \mathsf{F}\ \mathbf{else}\ \mathsf{U})$$

$$\texttt{forall3} \vdash_{\text{def}} \ddot{\forall} P = \mathbf{if}\ (\forall x.P(x) = \mathsf{T})\ \mathbf{then}\ \mathsf{T}\ \mathbf{else}\ (\mathbf{if}\ (\exists x.P(x) = \mathsf{F})\ \mathbf{then}\ \mathsf{F}\ \mathbf{else}\ \mathsf{U})$$

We use a two-valued theorem prover HOL [10] to reason about our geometry primitives. Introducing three-valued formulas into such a two-valued environment, poses the problem to integrate both logics. The conversion of three-valued expressions to Boolean domain depends on the proposition: In some situations, T should be the only designated truth value; in other cases, it suffices that a proposition P is 'at least U'. Although, this

$\leq$	F	U	T
F	T	T	T
U	F	T	T
T	F	F	T

$\geq$	F	U	T
F	T	F	F
U	T	T	F
T	T	T	T

$\twoheadrightarrow$	F	U	T
F	T	T	T
U	T	T	F
T	T	F	T

Fig. 3. Truth tables of $\leq$, $\geq$ and $\twoheadrightarrow$

can be expressed by $\neg(P = \mathsf{F})$, we introduce two new relations $\leq$ and $\geq$ to improve the readability. By their help, all relevant cases ($P = \mathsf{F}$, $P \leq \mathsf{U}$, $P \geq \mathsf{U}$), $P = \mathsf{T}$) can be described concisely (see Figure 3). The purpose of the operator $\twoheadrightarrow$ will be become clear in the following subsection.

2.2 Geometric Objects and Primitives

All geometric objects are formed by sets of points that are the solution of a proposition. To cope with endpoints and other extremal issues, we use three-valued inequations between rational numbers: for equal numbers, the validity of the inequation is U:

$$\texttt{les3} \vdash_{\text{def}} r_1 \prec r_2 = \mathbf{if}\,(r_1 < r_2)\,\mathbf{then}\,\mathsf{T}\,\mathbf{else}\,(\mathbf{if}\,(r_2 < r_1)\,\mathbf{then}\,\mathsf{F}\,\mathbf{else}\,\mathsf{U})$$

Using this relation, we define geometric objects. We thereby focus on two-dimensional linear objects, i.e. lines, segments and windows. Circles, curves, and objects of higher dimensions are not considered, since in embedded systems, they are usually approximated by linear objects.

A line is usually defined by its parametric equation. To convert the classic definition of a line to a three-valued one, all two-valued operators are exchanged by their three-valued counterparts ($\mathsf{beg}(\ell_1)$ and $\mathsf{end}(\ell_1)$ denote the two points that define the line):

$$\texttt{on_line_def} \vdash_{\text{def}} \mathsf{onLine}(\ell, v) = \ddot{\exists}\lambda.\, v = \mathsf{beg}(\ell) + \lambda \cdot (\mathsf{end}(\ell) - \mathsf{beg}(\ell))$$

For a line ℓ, there is no difference between the two-valued and three-valued case: ℓ contains all points $(x; y)$ that are a solution of the traditional, two-valued equation. For a line segment, λ must be greater than 0 and less than 1. With these restrictions, the end points are degenerate points.

$$\texttt{on_seg_def} \vdash_{\text{def}} \mathsf{onSeg}(\ell, v) = \ddot{\exists}\lambda.\, v = \mathsf{beg}(\ell) + \lambda \cdot (\mathsf{end}(\ell) - \mathsf{beg}(\ell)) \ddot{\wedge} (0 \prec \lambda) \ddot{\wedge} (\lambda \prec 1)$$

Most geometric algorithms rely on a small number of geometric primitives. Among them, there are primitives that take some input and classify it as one of a constant number of possible cases, as e.g.:

- *Position of two points.* A point p is left from a point q iff $\chi_{\mathsf{left}}(p, q) := x_q - x_p > 0$. Analogously, point p is below q iff $\chi_{\mathsf{below}}(p, q) := y_q - y_p > 0$.
- *Orientation of three points.* The points p, q and r define a left turn iff

$$\chi_{\mathsf{lturn}}(p, q, r) := \begin{vmatrix} x_p & y_p & 1 \\ x_q & y_q & 1 \\ x_r & y_r & 1 \end{vmatrix} > 0 \tag{1}$$

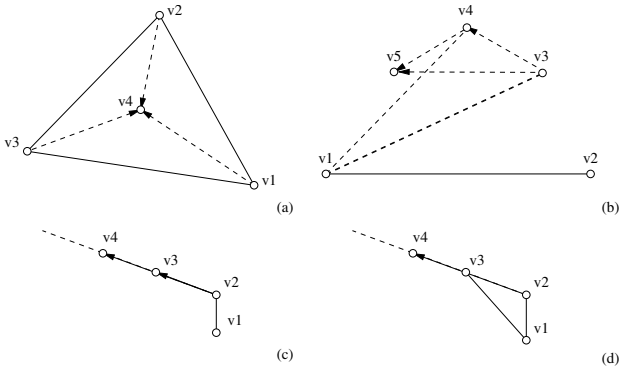


Fig. 4. Properties of the left-turn primitive

Degeneracies with respect to a such a primitive P are inputs $\boldsymbol{x}$ that cause the characteristic function to become zero $\chi_P(\boldsymbol{x}) = 0$. Following the approach presented in Section 2.1, the result U is returned in these cases. To define the primitives, we use the three-valued relation $\prec$ of the previous section. Since all primitives of the previous section compare their result with zero, we additionally introduce a relation pos :

$$\texttt{rat_pos} \vdash_{\text{def}} \mathsf{pos}\, r = 0 \prec r$$

With its help, the primitive *left* and *below* can be defined as follows:

$$\texttt{left} \vdash_{\text{def}} \mathsf{left}(v, w) = \mathsf{pos}\,(\mathsf{x}_w - \mathsf{x}_v)$$
$$\texttt{below} \vdash_{\text{def}} \mathsf{below}(v, w) = \mathsf{pos}\,(\mathsf{y}_w - \mathsf{y}_v)$$

The primitives make use of the three-valued relation $\prec$, and thus, they have similar properties: The following theorems prove some sort of reflexivity, antisymmetry, and transitivity laws.

$$\texttt{LEFT_REF} \vdash \mathsf{left}(v, v) = \mathsf{U}$$
$$\texttt{LEFT_ASYM} \vdash \mathsf{left}(v, w) = \ddot{\neg}\mathsf{left}(w, v)$$
$$\texttt{LEFT_TRANS} \vdash \mathsf{left}(u, v) \mathbin{\ddot{*}} \mathsf{left}(v, w) \twoheadrightarrow \mathsf{left}(u, w)$$

LEFT_TRANS makes use of the connectives $\ddot{*}$ and $\twoheadrightarrow$, which usually appear together in a proposition. They allow a succinct description of the following cases:

- If $\mathsf{left}(u, v) = \mathsf{T}$ and $\mathsf{left}(v, w) = \mathsf{T}$, then $\mathsf{left}(u, w) = \mathsf{T}$.
- If $\mathsf{left}(u, v) = \mathsf{T}$ and $\mathsf{left}(v, w) = \mathsf{U}$ or vice versa, then $\mathsf{left}(u, w) = \mathsf{T}$).
- If $\mathsf{left}(u, v) = \mathsf{U}$ and $\mathsf{left}(v, w) = \mathsf{U}$, then $\mathsf{left}(u, w) = \mathsf{U}$.
- If $\mathsf{left}(u, v) = \mathsf{F}$ or $\mathsf{left}(v, w) = \mathsf{F}$, then nothing is said about $\mathsf{left}(u, w)$.

The orientation primitives can be analogously defined:

$$\texttt{lturn} \vdash_{\text{def}} \mathsf{lturn}(u, v, w) = \mathsf{pos}\,((v - u) \times (w - v))$$
$$\texttt{rturn} \vdash_{\text{def}} \mathsf{rturn}(u, v, w) = \mathsf{lturn}(w, v, u)$$

Again, various properties can be proven for the orientation primitive:

$$\texttt{LTURN_REF} \vdash \mathsf{lturn}(v_1, v_1, v_2) = \mathsf{U}$$
$$\texttt{LTURN_SYM} \vdash \mathsf{lturn}(v_1, v_2, v_3) = \mathsf{lturn}(v_2, v_3, v_1)$$
$$\texttt{LTURN_ASYM} \vdash \mathsf{lturn}(v_1, v_2, v_3) = \ddot{\neg}\mathsf{lturn}(v_2, v_1, v_3)$$
$$\texttt{LTURN_TRIAN} \vdash$$
$$\mathsf{lturn}(v_1, v_2, v_4) \ddot{*} \mathsf{lturn}(v_2, v_3, v_4) \ddot{*} \mathsf{lturn}(v_3, v_1, v_4) \twoheadrightarrow \mathsf{lturn}(v_1, v_2, v_3)$$
$$\texttt{LTURN_TRANS} \vdash (\mathsf{lturn}(v_1, v_2, v_3) \ddot{\wedge} \mathsf{lturn}(v_1, v_2, v_4) \ddot{\wedge} \mathsf{lturn}(v_1, v_2, v_5) \geq \mathsf{U})$$
$$\Rightarrow \mathsf{lturn}(v_1, v_3, v_4) \ddot{*} \mathsf{lturn}(v_1, v_4, v_5) \twoheadrightarrow \mathsf{lturn}(v_1, v_3, v_5)$$
$$\texttt{LTURN_MOD1} \vdash (\mathsf{onRay}(\overrightarrow{(v_2, v_3)}, v_4) = \mathsf{T}) \Rightarrow \mathsf{lturn}(v_1, v_2, v_3) = \mathsf{lturn}(v_1, v_2, v_4)$$
$$\texttt{LTURN_MOD2} \vdash (\mathsf{onRay}(\overrightarrow{(v_4, v_3)}, v_2) = \mathsf{T}) \Rightarrow \mathsf{lturn}(v_1, v_2, v_4) = \mathsf{lturn}(v_1, v_3, v_4)$$

These theorems are three-valued reformulations of the ones that can be found in [15]. The first three theorems (LTURN_REF, LTURN_SYM and LTURN_ASYM) state that a sequence in which a point appears at least twice is a degenerate case. Moreover, a sequence can be rotated without changing the orientation, and two points can be interchanged with negating the orientation of the sequence. LTURN_TRIAN describes the situation depicted in Figure 4 (a): If a point is on the positive side of three pairwise connected segments, they form a triangle with positive orientation. LTURN_TRANS proves the transitivity of the left-turn primitive under the condition that the three points v_3, v_4 and v_5 lie on the positive side of a segment from v_1 to v_2 (see Figure 4 (b)). The last two theorems (Figure 4 (c) and (d)) are used in [15] to handle degenerate cases. Actually, they are not needed in our approach, since LTURN_TRIAN already covers these cases. This illustrates the advantages of our approach: We always address general and degenerate cases at the same time, which makes the description succinct and readable. The same holds for later implementations that are made with three-valued data types.

3 Cohen-Sutherland Line Clipping

3.1 Formalisation of the Algorithm in HOL

A frequent operation in many graphic applications is the line clipping to an upright rectangular window. To this end, the Cohen-Sutherland algorithm divides the plane into nine regions: Each of the edges of the clip window defines an infinite line that divides the plane into inside and outside half-spaces (see Figure 5 (a)). The resulting regions can be described using a four bit *outcode*. The four bits of this code denote whether this region is situated *above*, *below*, *left* and *right* of the window, respectively

With the help of this outcode, necessary conditions can be formulated to check whether the line segment is inside or outside the window: The line segment is inside the window iff no bit of the outcodes of the endpoints is set. If a common bit is set in the outcodes of the endpoints, then both endpoints are outside the same side of the window. Thus, the entire line segment is outside the window. In most cases, a line segment will have been either accepted of rejected by these conditions.

In all other cases, the line segment is split into two pieces at an appropriate clipping edge. Assume that the considered end point (x_1, y_1) is above the window (see Figure 5 (b)). Removing the portion of the line that is above the window results in a new line segment with the old endpoint (x_2, y_2) and the new endpoint (x'_1, y'_1). Since the new

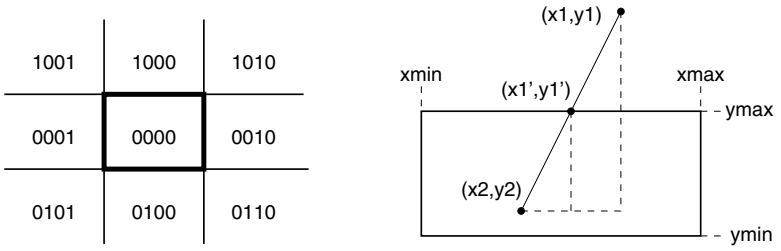


Fig. 5. Cohen-Sutherland Algorithm: (a) *Outcodes*, (b) Computing the intersection

endpoint is on the top border of the window, we have $y_1' = y_{\max}$. The other coordinate x_1' can be computed as follows:

$$x_1' = x_1 + (x_2 - x_1) \cdot \frac{y_{\max} - y_1}{y_2 - y_1}.$$

Once the line segment is identified, the outcode of the new endpoint is computed. After this, the algorithm is restarted with the new values.

In order to formalise the algorithm in HOL, we start with the definition of the outcodes. To this end, we use the primitives $\mathsf{lole}(w)$ and $\mathsf{upri}(w)$, which denote the lower left and the upper right corner of the window w:

$$\texttt{outcode} \vdash_{\text{def}} \mathsf{code}(w, v) = (\mathsf{below}(v, \mathsf{lole}(w)) = \mathsf{T},\ \mathsf{above}(v, \mathsf{upri}(w)) = \mathsf{T},\ \mathsf{left}(v, \mathsf{lole}(w)) = \mathsf{T},\ \mathsf{right}(v, \mathsf{upri}(w)) = \mathsf{T})$$

$$\texttt{INSIDE} \vdash_{\text{def}} \mathsf{INSIDE}(c_1) = (c_1 = (\mathsf{F}, \mathsf{F}, \mathsf{F}, \mathsf{F}))$$

$$\texttt{BOTTOM} \vdash_{\text{def}} \mathsf{BOTTOM}(c_1) = c_1[0]$$

$$\texttt{TOP} \vdash_{\text{def}} \mathsf{TOP}(c_1) = c_1[1]$$

$$\texttt{LEFT} \vdash_{\text{def}} \mathsf{LEFT}(c_1) = c_1[2]$$

$$\texttt{RIGHT} \vdash_{\text{def}} \mathsf{RIGHT}(c_1) = c_1[3]$$

$$\texttt{ACCEPT} \vdash_{\text{def}} \mathsf{ACCEPT}(c_1, c_2) = \neg(c_1[0] \vee c_2[0] \vee c_1[1] \vee c_2[1] \vee c_1[2] \vee c_2[2] \vee c_1[3] \vee c_2[3])$$

$$\texttt{REJECT} \vdash_{\text{def}} \mathsf{REJECT}(c_1, c_2) = c_1[0] \wedge c_2[0] \vee c_1[1] \wedge c_2[1] \vee c_1[2] \wedge c_2[2] \vee c_1[3] \wedge c_2[3]$$

The actual algorithm is taken from [9], where the loop is replaced by a recursive call. Moreover, as the algorithm does not return an edge in all cases, an *option type* is used for the result, returning NONE if the line is rejected, and $\mathsf{SOME}(e)$ if the edge e is accepted.

$$\begin{aligned}
&\texttt{csa_clip} \vdash_{\text{def}} \mathsf{csaClip}(w, (v_{\mathrm{b}}, v_{\mathrm{e}})) = \\
&\quad \textbf{if } \mathsf{ACCEPT}(\mathsf{code}(w, v_{\mathrm{b}}), \mathsf{code}(w, v_{\mathrm{e}})) \textbf{ then} \\
&\qquad \mathsf{SOME}((v_{\mathrm{b}}, v_{\mathrm{e}})) \\
&\quad \textbf{else if } \mathsf{REJECT}(\mathsf{code}(w, v_{\mathrm{b}}), \mathsf{code}(w, v_{\mathrm{e}})) \textbf{ then} \\
&\qquad \mathsf{NONE} \\
&\quad \textbf{else if } \mathsf{INSIDE}(\mathsf{code}(w, v_{\mathrm{b}})) \textbf{ then} \\
&\qquad \mathsf{csaClip}(w, (v_{\mathrm{e}}, v_{\mathrm{b}})) \\
&\quad \textbf{else} \\
&\qquad \mathsf{csaClip}(w, \mathsf{shortenedLine}(w, (v_{\mathrm{b}}, v_{\mathrm{e}})))
\end{aligned}$$

$$
\begin{array}{l}
\texttt{shortened_line} \vdash_{\text{def}} \mathsf{shortenedLine}(w, (v_\text{b}, v_\text{e})) = \\
\quad \textbf{let} \\
\qquad x_\text{min} = \mathsf{x}_{\mathsf{lole}(w)} \textbf{ and } y_\text{min} = \mathsf{y}_{\mathsf{lole}(w)} \textbf{ and} \\
\qquad x_\text{max} = \mathsf{x}_{\mathsf{upri}(w)} \textbf{ and } y_\text{max} = \mathsf{y}_{\mathsf{upri}(w)} \textbf{ and} \\
\qquad x_1 = \mathsf{x}_{v_\text{b}} \textbf{ and } y_1 = \mathsf{y}_{v_\text{b}} \textbf{ and } x_2 = \mathsf{x}_{vE} \textbf{ and } y_2 = \mathsf{y}_{v_\text{e}} \textbf{ and} \\
\qquad c_1 = \mathsf{code}(w, v_\text{b}) \\
\quad \textbf{in} \\
\qquad \textbf{if } \mathsf{TOP}(c_1) \textbf{ then} \\
\qquad\quad \left(\left(x_1 + (x_2 - x_1) \cdot \dfrac{y_\text{max} - y_1}{y_2 - y_1} \,;\, y_\text{max} \right), v_\text{e} \right) \\
\qquad \textbf{else if } \mathsf{BOTTOM}(c_1) \textbf{ then} \\
\qquad\quad \left(\left(x_1 + (x_2 - x_1) \cdot \dfrac{y_\text{min} - y_1}{y_2 - y_1} \,;\, y_\text{min} \right), v_\text{e} \right) \\
\qquad \textbf{else if } \mathsf{LEFT}(c_1) \textbf{ then} \\
\qquad\quad \left(\left(x_\text{min} \,;\, y_1 + (y_2 - y_1) \cdot \dfrac{x_\text{min} - x_1}{x_2 - x_1} \right), v_\text{e} \right) \\
\qquad \textbf{else} \\
\qquad\quad \left(\left(x_\text{max} \,;\, y_1 + (y_2 - y_1) \cdot \dfrac{x_\text{max} - x_1}{x_2 - x_1} \right), v_\text{e} \right)
\end{array}
$$

3.2 Verification

As a first step of the verification, we have to set up a formal specification of the algorithm: Given a line segment (by its two endpoints v_b and v_e) and a rectangular window (given by its lower left and its upper right corners), return the line segment that is inside the window, where all points on the edge are considered to be inside. To keep the specification concise, we use the following two primitives: $\mathsf{csaInWin}(w_1, v)$ holds if the point v is inside or on the edge of window w_1, and $\mathsf{csaOnSeg}((v_\text{b}, v_\text{e}), v)$ holds if the point v is on the line segment from v_b to v_e (including the endpoints). As before, $\mathsf{lole}(w_1)$ and $\mathsf{upri}(w_1)$ denote the lower left and upper right corner of the clipping window. Note that degenerate inputs and outputs are possible, since the input and the output segments may consist of a single point.

$$
\begin{array}{l}
\texttt{csa_in_win} \vdash_{\text{def}} \mathsf{csaInWin}(w_1, v) = \mathsf{inWin}(\mathsf{lole}(w_1), \mathsf{upri}(w_1), v) \geq \mathsf{U} \\
\texttt{csa_on_seg} \vdash_{\text{def}} \mathsf{csaOnSeg}((v_\text{b}, v_\text{e}), v) = \\
\quad \textbf{if } (v_\text{b} = v_\text{e}) \textbf{ then } (v_\text{b} = v) \textbf{ else } \mathsf{onSeg}(\overrightarrow{(v_\text{b}, v_\text{e})}, v) \geq \mathsf{U})
\end{array}
$$

In any case, the result is a value of an option type: it is either NONE (if the line segment is outside the window) or it represents a pair of points defining the clipped line segment.

$$
\begin{array}{l}
\texttt{CSA_CORRECTNESS} \vdash \\
\quad \textbf{let } r = \mathsf{csaClip}(w_1, (v_\text{b}, v_\text{e})) \textbf{ in} \\
\qquad \textbf{if } (\mathsf{IS_NONE}(r)) \textbf{ then} \\
\qquad\quad \neg(\exists v.\, \mathsf{csaOnSeg}((v_\text{b}, v_\text{e}), v) \wedge \mathsf{csaInWin}(w_1, v)) \\
\qquad \textbf{else} \\
\qquad\quad \forall v.\, \mathsf{csaOnSeg}(\mathsf{THE}(r), v) = (\mathsf{csaOnSeg}((v_\text{b}, v_\text{e}), v) \wedge \mathsf{csaInWin}(w_1, v))
\end{array}
$$

The correctness of the above specification is proven by induction over the recursive calls of the clipping function $\mathsf{csaClip}(w_1, (v_\text{b}, v_\text{e}))$. There are two base cases: In the first case,

the line segment is accepted, since both endpoints are inside the window. We must prove that every point of the segment is also in the window (ACCEPT_SEGMENT_INWINDOW).

$$\texttt{ACCEPT_SEGMENT_INWIN} \vdash \mathsf{ACCEPT}(\mathsf{code}(w_1, v_\mathrm{b}), \mathsf{code}(w_1, v_\mathrm{e})) \Rightarrow \mathsf{csaOnSeg}((v_\mathrm{b}, v_\mathrm{e}), v) \Rightarrow \mathsf{csaInWin}(w_1, v)$$

The second base case results from the rejection of the line segment in the first recursive call. For the proof, we use the bounding box check: If the reject primitive holds, we prove that the window that is defined by the endpoints of the line segment and the clipping window have no common area. Thus, the segment does not have an intersection with the clipping window, and it is correctly rejected.

$$\begin{aligned}&\texttt{bbox_check} \vdash_{\mathrm{def}} \mathsf{bboxCheck}(v_1, v_2, v_3, v_4) = \\ &\quad (\mathsf{vecXint}(v_1, v_3, v_2) \ddot{\vee} \mathsf{vecXint}(v_1, v_4, v_2) \ddot{\vee} \\ &\quad\ \mathsf{vecXint}(v_3, v_1, v_4) \ddot{\vee} \mathsf{vecXint}(v_3, v_2, v_4) \geq \mathsf{U}) \wedge \\ &\quad (\mathsf{vecYint}(v_1, v_3, v_2) \ddot{\vee} \mathsf{vecYint}(v_1, v_4, v_2) \ddot{\vee} \\ &\quad\ \mathsf{vecYint}(v_3, v_1, v_4) \ddot{\vee} \mathsf{vecYint}(v_3, v_2, v_4) \geq \mathsf{U})\end{aligned}$$

$$\texttt{REJECT_BBOX_CHECK} \vdash \mathsf{REJECT}(\mathsf{code}(w_1, v_\mathrm{b}), \mathsf{code}(w_1, v_\mathrm{e})) \Rightarrow \neg\mathsf{bboxCheck}(\mathsf{lole}(w_1), \mathsf{upri}(w_1), v_\mathrm{b}, v_\mathrm{e})$$

$$\texttt{CSA_BBOX_CHECK} \vdash \neg(\mathsf{bboxCheck}(v_1, v_2, v_3, v_4) \wedge (\mathsf{inWin}(v_3, v_4, v) \geq \mathsf{U}) \Rightarrow \neg(\mathsf{inWin}(v_1, v_2, v) \geq \mathsf{U})$$

The first recursive call swaps the end points. Provided that the induction hypotheses holds, the algorithm is correct, because swapping the end points does not change the set of points of the segment (CSA_ONSEG_SYM).

$$\texttt{CSA_ONSEG_SYM} \vdash \mathsf{csaOnSeg}((v_\mathrm{b}, v_\mathrm{e}), v) = \mathsf{csaOnSeg}((v_\mathrm{e}, v_\mathrm{b}), v)$$

The second recursive call is the hardest part of the proof. The line segment is shortened. Two things must be proven: First, the shortened segment is a subset of the original one (CSA_CUT_CORRECT). Second, points on the segment that are inside the window are not cut off (CSA_CUT_COMPLETE).

$$\begin{aligned}&\texttt{CSA_CUT_CORRECT} \vdash \\ &\quad \neg\mathsf{ACCEPT}(\mathsf{code}(w_1, v_\mathrm{b}), \mathsf{code}(w_1, v_\mathrm{e})) \wedge \\ &\quad \neg\mathsf{REJECT}(\mathsf{code}(w_1, v_\mathrm{b}), \mathsf{code}(w_1, v_\mathrm{e})) \wedge \\ &\quad \neg\mathsf{INSIDE}(\mathsf{code}(w_1, v_\mathrm{b})) \wedge \\ &\quad \mathsf{csaInWin}(w_1, v) \Rightarrow \\ &\quad \mathsf{csaOnSeg}((\mathsf{shortenedLine}(w_1, (v_\mathrm{b}, v_\mathrm{e}))), v) \Rightarrow \mathsf{csaOnSeg}((v_\mathrm{b}, v_\mathrm{e}), v)\end{aligned}$$

$$\begin{aligned}&\texttt{CSA_CUT_COMPLETE} \vdash \\ &\quad \neg\mathsf{ACCEPT}(\mathsf{code}(w_1, v_\mathrm{b}), \mathsf{code}(w_1, v_\mathrm{e})) \wedge \\ &\quad \neg\mathsf{REJECT}(\mathsf{code}(w_1, v_\mathrm{b}), \mathsf{code}(w_1, v_\mathrm{e})) \wedge \\ &\quad \neg\mathsf{INSIDE}(\mathsf{code}(w_1, v_\mathrm{b})) \wedge \\ &\quad \mathsf{csaInWin}(w_1, v) \Rightarrow \\ &\quad \mathsf{csaOnSeg}((v_\mathrm{b}, v_\mathrm{e}), v) \Rightarrow \mathsf{csaOnSeg}(\mathsf{shortenedLine}(w_1, (v_\mathrm{b}, v_\mathrm{e}), v)\end{aligned}$$

$$\begin{aligned}&\texttt{LINE_SPLIT} \vdash (\mathsf{onSeg}(\ell_1, v_m) = \mathsf{T}) \Rightarrow \\ &\quad ((\mathsf{onSeg}(\ell_1, v) \geq \mathsf{U}) = (v = \mathsf{beg}(\ell_1)) \vee (\mathsf{onSeg}(\overrightarrow{(\mathsf{beg}(\ell_1), v_m)}, v) = \mathsf{T}) \vee \\ &\qquad\qquad (\mathsf{onSeg}(\overrightarrow{(v_m, \mathsf{end}(\ell_1))}, v) \geq \mathsf{U}))\end{aligned}$$

The key to prove the remaining part is to show that the line segment is split into two partitions (LINE_SPLIT). This concludes the correctness proof that holds for all cases

(including that endpoints are on the edges of the window, or that both endpoints are the same). Hence, we have proven that the algorithm terminates and that it returns the specified result.

4 Implementation of a Dependable Software Library

The previous sections illustrated how rational numbers and a three-valued logic can be used to specify and verify polygon processing algorithms. We use the same techniques to implement a software library. The advantages are obvious: First, the specification and the actual implementation are as close as possible, since all primitives are implemented in software in the same way as they were defined in the HOL theory. Second and more important, algorithms are more compact, because several cases of the two-valued formalisation can be merged in the three-valued setting.

We implemented a software library in C based on three-valued logics and rational numbers. With the help of the GMP library, we perform all calculations with arbitrary precision. The following paragraphs sketch the implementation.

The truth values of the three valued logic are represented by a signed integer, and the logical functions are implemented by their truth tables.

```
typedef signed char log3 /* F=-1, U=0, T=1 */;

const log3 not3_table[] = {T,U,F};
log3 not3( log3 a ) { return not3_table[a+1] }

const log3 and3_table[] = {{F,F,F},{F,U,U},{F,U,T}};
log3 and3( log3 a, log3b ) { return and3_table[a+1][b+1] }
```

The existence of an intersection point of two line segments can be computed as follows:

```
log3 do_seg_intersect ( line l1, line l2 ) {
  return and3(
   equ3(
    lturn( l1.beg,l1.end,l2.beg ), rturn( l1.beg,l1.end,l2.end )
   ),(
    lturn( l2.beg,l2.end,l1.beg ), rturn( l2.beg,l2.end,l1.end )
   ))
}
```

If both segments truly intersect, T is returned. If they only touch or have a common line segment, U is returned, since this is a degenerated case. If there are no common points, F is returned. This corresponds to the function $\ddot{\exists} x.\, \mathsf{onSeg}(\ell_1, x) \ddot{\wedge} \mathsf{onSeg}(\ell_2, x)$.

As another example, reconsider the *winding number* algorithm, which is described in Section 2.1. A classic implementation of this algorithm that directly deals with degenerate cases cannot avoid the case distinctions shown in Figure 1. The position of the previous and following points must be taken into account, which leads to even more subcases. With the help of a three-valued intersection, the algorithm can be formulated much easier. Three-valued primitives eliminate all extrinsic degeneracies. This clearly demonstrates the benefits of our approach.

Provided that `e[i]` denotes the i-th of n edges of a polygon and `xRay(v)` is the ray from v to the right, the following C fragment calculates the winding number w of v.

```
w = 0;
for( i = 0 ; i < n ; i++ )
if( doIntersect( xRay(v),e[i] ) >= U )
{
    w += below( v, e[i].begin );
    w += above( v, e[i].end );
}
w = w / 2;
```

5 Conclusions

In this paper, we propose the use of three-valued logic to develop dependable algorithms for geometric applications to be used in safety-critical embedded systems. The use of a third truth value allows us to make degenerate cases explicit so that they can be appropriately handled by different algorithms. Starting from applications like motion planning and collision detection, we dealt with basic geometric objects and primitives.

The obtained software library has been formalised in the interactive theorem prover HOL. Moreover, we used HOL to formally reason about the geometric primitives in order to assure their consistent use. In particular, we are able to verify entire algorithms that are used in embedded systems in order to guarantee that required safety properties like avoidance of collisions are met.

Furthermore, since degenerate cases are succinctly described by three-valued primitives, the derived algorithms are both robust and compact, and their results are to a large extent independent of the numeric precision of the underlying microprocessors.

References

1. M. Alt, C. Ferdinand, F. Martin, and R. Wilhelm. Cache behavior prediction by abstract interpretation. In *Static Analysis Symposium (SAS)*, volume 1145 of *LNCS*, pages 52–66, Aachen, Germany, 1996. Springer.
2. G. Berry. The constructive semantics of pure Esterel. http://www-sop.inria.fr/esterel.org, July 1999.
3. J. Brandt and K. Schneider. Using three-valued logic to specify and verify algorithms of computational geometry. In *International Conference on Formal Engineering Methods (ICFEM)*, LNCS, Manchester, UK, 2005. Springer.
4. J.A. Brzozowski and C.-J.H. Seger. *Asynchronous Circuits*. Springer, 1995.
5. C. Burnikel, K. Mehlhorn, and S. Schirra. On degeneracy in geometric computations. In *Symposium on Discrete Algorithms (SODA)*, pages 16–23, Arlington, Virginia, USA, 1994. ACM.
6. S.C. Chou, X.S. Gao, and J.Z. Zhang. *Machine Proofs in Geometry*. World Scientific, Singapore, 1994.
7. M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry*. Springer, 2000.
8. H. Edelsbrunner and E.P. Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 9(1):66–104, 1990.

9. J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes. *Computer Graphics: Principles and Practice*. Addison Wesley, 2000.
10. M.J.C. Gordon and T.F. Melham. *Introduction to HOL: A Theorem Proving Environment for Higher Order Logic*. Cambridge University Press, 1993.
11. K. Hormann and A. Agathos. The point in polygon problem for arbitrary polygons. *Computational Geometry*, 20(3):131–144, 2001.
12. S.C. Kleene. *Introduction to Metamathematics*. North Holland, 1952.
13. S. Malik. Analysis of cyclic combinational circuits. In *Conference on Computer Aided Design (ICCAD)*, pages 618–625, Santa Clara, California, November 1993. IEEE Computer Society.
14. K. Mehlhorn and S. Näher. *The LEDA Platform of Combinatorial and Geometric Computing*. Cambridge University Press, 1999.
15. D. Pichardie and Y. Bertot. Formalizing convex hull algorithms. In R.J. Boulton and P.B. Jackson, editors, *Higher Order Logic Theorem Proving and its Applications (TPHOL)*, volume 2152 of *LNCS*, pages 346–361, Edinburgh, Scotland, UK, 2001. Springer.
16. T.W. Reps, M. Sagiv, and R. Wilhelm. Static program analysis via 3-valued logic. In R. Alur and D.A. Peled, editors, *Conference on Computer Aided Verification (CAV)*, volume 3114 of *LNCS*, pages 15–30, Boston, MA, USA, 2004. Springer.
17. K. Schneider, J. Brandt, T. Schuele, and T. Tuerk. Improving constructiveness in code generators. In *Synchronous Languages, Applications, and Programming (SLAP)*, Edinburgh, Scotland, UK, 2005.
18. K. Schneider, J. Brandt, T. Schuele, and T. Tuerk. Maximal causality analysis. In *Conference on Application of Concurrency to System Design (ACSD)*, pages 106–115, St. Malo, France, June 2005. IEEE Computer Society.
19. T. Schuele and K. Schneider. Three-valued logic in bounded model checking. In *Formal Methods and Models for Codesign (MEMOCODE)*, Verona, Italy, 2005. IEEE Computer Society.
20. T.R. Shiple. *Formal Analysis of Synchronous Circuits*. PhD thesis, University of California at Berkeley, 1996.
21. W.-T. Wu. On the decision problem and the mechanization of theorem proving in elementary geometry. *Scientia Sinica*, 21:157–179, 1978.

New Area Management Method Based on "Pressure" for Plastic Cell Architecture

Taichi Nagamoto, Satoshi Yano, Mitsuru Uchida,
Yuichiro Shibata, and Kiyoshi Oguri

Department of Electrical Engineering and Computer Science,
Graduate School of Science and Technology, Nagasaki University,
1–14 Bunkyo-machi, Nagasaki 852-8521, Japan
{taichi, yano, uchida}@pca.cis.nagasaki-u.ac.jp
{shibata, oguri}@cis.nagasaki-u.ac.jp

Abstract. In the present paper, we propose a novel area management method based on the concept of "pressure". Plastic Cell Architecture (PCA) is a dynamically reconfigurable architecture that was proposed paying attention to the essentials of processing and the flexibility of Von Neumann architecture. Mechanisms in which area are managed require a function similar to "malloc" in the C language. However, for dynamically reconfigurable architectures such as PCA, uniform management and parallelism are incompatible. Therefore, it is necessary for any PCA circuit to determine its own vacant areas without assistance from an administrator. We therefore introduced a new area management method that obtains the vacant areas by moving the surrounding objects by "pressure". "Pressure" is realize by adding new command set to original commands of PCA. We describe the basic structure for the new command set, and consider herein the development of three new command sets. Finally, we evaluate these command sets with respect to execution area and rate of effective use.

1 Introduction

In the age of ubiquitous computing, the von Neumann type computer effects every aspect of life and has infiltrated every corner of our society. The key to the success of the von Neumann type computer is its generality. We believe that the architecture of the hardware and the software (i.e. program) drive the generality. Any function can be realized both in program logic and wired logic. Important difference between correct program logic and wired logic is that only program logic can allow dynamical constitution with an object instance during execution of a procedure. This heap management is thought to be the source of the pliability of the CPU + MEM structure. Therefore, heap management should also be considered in the domain of wired logic.

Plastic Cell Architecture (PCA) is a new computer architecture having a high degree of parallelism that can replaces the von Neumann type architecture. PCA was designed taking flexibility in wired logic domain [1], [2]. PCA has a structure in

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 418–427, 2005.
© IFIP International Federation for Information Processing 2005

which LUTs (Look-Up Tables), which are small-scale truth value tables, are set into the routing network. PCA can perform "processing, transmission, and holding" of data by using the LUTs as memory or as a circuit and the routing network.

In order to use enable dynamic-reconfigurability, which is characteristic of PCA, a functions such as "malloc" and "new", from the C/C++ language, are required. PCA is not yet provided with a domain management method for using these functions because of a problem concerning placement of a new instance. For example, there exists a method of using "domain administrator" that manages areas, but management and the parallel system are incompatible. Furthermore, access to the domain administrator may increase the processing overhead, causing deadlock.

In this paper, we propose a area management method that uses the concept of "pressure". This method is based on a cell division mechanism, whereby existing cells are pushed out of the way when by the growth of a new cell. In this case, the "cell" is the new instance. This method can be performed independently in several places, so that domain administrator is not necessary. PCA achieves dynamic-reconfigurability using commands that are implemented in the built-in part (see Section 2). Therefore, this method is realized as a extension of commands in the built-in part.

In Section 2, we present the background and the related work. In Section 3, we propose a new area management scheme, and in Section 4, we describe the mechanism required to create a command set. In Section 5, three command sets are introduced, and these command sets are evaluated in Section 6. Finally, conclusions are presented in Section 7.

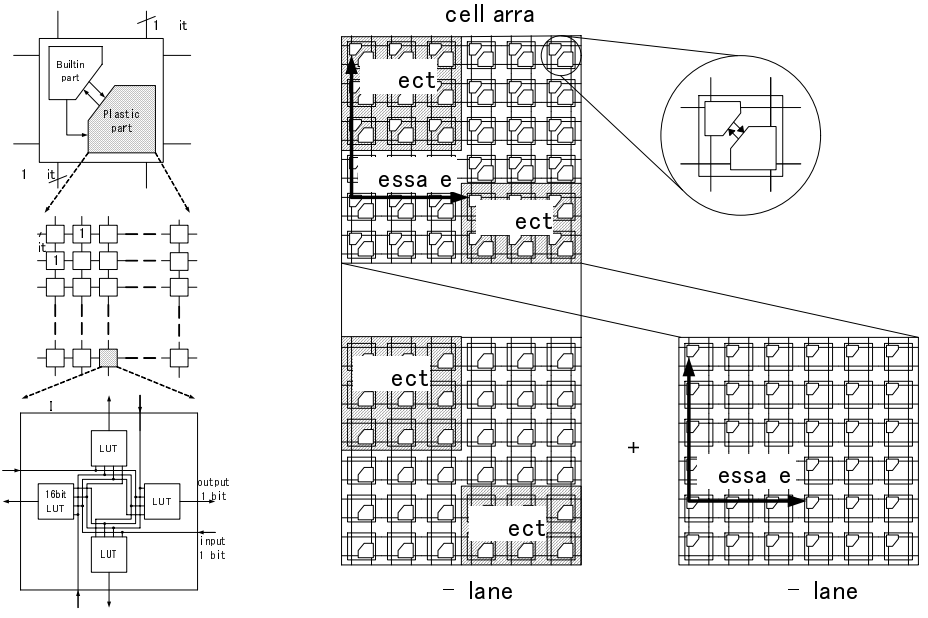
2 Background

2.1 PCA (Plastic Cell Architecture)

The composition of PCA is covered with LUTs that are in the routing network. In this composition, data processing, transmission, and memory can be performed directly and in parallel[1], [2].

Figure 1(a) shows the basic structure of PCA, which is referred to as the PCA cell. PCA cells are arranged in a two-dimensional mesh. Each PCA cell consists of a plastic part and a built-in part. Plastic parts are reconfigurable circuits that can perform memory and operation tasks, and built-in parts performs control and communication tasks for each of the plastic parts. Plastic parts have 8×8 basic cells. The basic cell consists of four 16-bit LUTs. An LUT has a memory of 4 bits for address input and 1 bit for data output.

Figure 1(b) shows an example of communication between objects. An aggregate of PCA cells that performs processing is referred to as an object. Objects that become elements (e.g. add, sub, conditional branch) of a process are arranged to correspond to a Data Flow Diagram. Communication between objects is performed by a built-in part. When two or more objects send messages to each other, the built-in part operates in parallel. Furthermore, an object can be dynamically created by sending a composition message to the built-in part.



(a) PCA Cell (b) Object and Message

Fig. 1. Plastic Cell Architecture

In PCA, an asynchronous circuit system with easy timing control at the time of connecting a new object is adopted. PCA adopts a bundled protocol with four-cycle signaling.

2.2 Bit Serial PCA

An architecture called Bit Serial PCA, based on PCA has been proposed in a previous study [3]. The characteristics of Bit Serial PCA are described below.

There are two types of data-processing system: the bit parallel system and the bit serial system. Although the bit parallel system is most commonly used at present, if the skew between signals and the rate of wiring delay increase, then the bit serial system may provide better high-speed performance than the bit parallel system. In addition, since PCA hardware can be reconfigured, a switch for reconfiguration exists in the wiring, further increasing the overhead. PCA also arranges several operation circuits along the data flow and performs arbitrary processing. Therefore, the bit serial system with small circuits for each function is suitable for PCA. Therefore, Bit Serial PCA adopts the bit serial system. As a result, circuits can be constituted only with state machines and shift registers, allowing the circuit to be compact while offering high-speed performance.

Each plastic part operates as a state machine and a shift register, and communication of plastic parts is performed through a built-in part.

2.3 Related Work

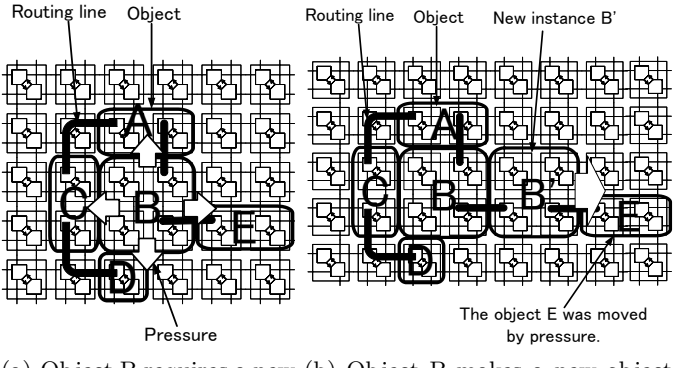
Interest in dynamic reconfigurable technology has been increasing in recent years. Replacing hardware having pliability with exclusive hardware is expected to enable various applicable fields on one chip [4].

As methods that realize a dynamically reconfigurable system, the method of changing functions and connection of a composition element is examined. The multi-context system uses a method that changes two or more pieces of composition information (commands) stored in the internal memory if needed. Multi-context type reconfigurable devices were initially proposed for reconfigurable devices to examine the degree of granulation. For example, at Keio University, a performance improvement was achieved by using a variable clock [5]. The DRP, developed by NEC, is another example of a multi-context type reconfigurable device [6].

NTT also proposed a PCA system [7], [8], [9]. Furthermore, a prototype PCA-2, which combines a high degree of integration and high speed, has been developed [10]. PCA-2 has a function whereby its composition can be partially changed, based on the data generated by the PCA objects themselves. In PCA, the composition of circuits fluctuates somewhat. In order to employ the reconfigurable function of PCA efficiently, the problem of how the domain of circuit should be constituted is encountered. At the University of Aizu, a method of expressing the entire circuit by a unit that includes duplicates or deletions, called a stage, was proposed [11].

3 New Method of Area Management

The concept of PCA is shown in Fig. 2. The squares in the figure indicate objects, and the thick lines indicate routing lines. Each object consists of one or more PCA cell. Each object works as a circuit. Each object performs an



(a) Object B requires a new object (b) Object B makes a new object B'

Fig. 2. Concept of PCA

operation (e.g. add, sub, conditional branch). Communication between objects is performed through the built-in parts. A new method of area management is shown below.

When object B needs a new object B', it sends a "pressure command" to the surrounding objects (Fig. 2(a)). The surrounding objects receive the pressure command and move to vacant areas. (In the figure, object E moved (Fig. 2(b)).) If the area is vacant, object B makes a new object (B'). This mechanism is realized as an extension of the command in the built-in part.

In clarifying the characteristics and the effect of area management caused by the pressure, we assumed the following in order to simplify the problems.

1. Routing lines between objects are not considered.
2. An object consists of one PCA cell.

There are numerous command sets that would allow us to realize this method. However, creating the correct command set is difficult. In this paper, we present three types of correct command sets having different characteristics. These command sets were obtained by trial and error.

4 Mechanism Required in Order to Create a Command Set

In order to create a command set, we implemented a simulator on a PC and performed trial and error repeatedly. As a result, the mechanism shown in Fig. 3 was found to be necessary. Cells are arrayed onto 2D meshes, as in PCA, to process pressure commands (Fig. 3 left). Each cell consists of a state machine to identify the condition (e.g. vacant state, circuit state, or receiving a pressure state). Cells have five input-output ports (north, south, east, west and feedback), and a switch to limit the direction of entered commands (Fig. 3 right).

Communication Path for Commands. In communication between cells, deadlock is a very important problem. The structure buffer and e-cube routing, for example, are reported to be communicative methods that do not incur deadlock. However, these method are expensive to implement on PCA hardware. When deadlock occurs, the sender must wait until a receiver becomes

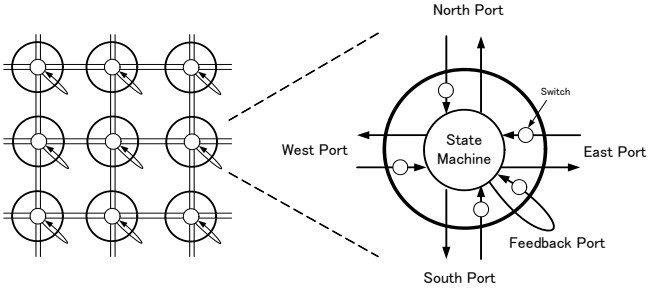


Fig. 3. Cells

available. Therefore, we adopted a communication system that does not wait for responses. The system is based on wormhole routing and can output commands without waiting for a response from the receiver side, even if a cell that receives a command is processing. Although this system prevents deadlock, the possibility exists that commands will be overwritten.

Function to Limit the Input Port. When a cell moved by pressure duplicates, a contradiction occurs if this information is not correctly communicated. Therefore, in order to transmit the information correctly, we added a function whereby the cells do not process commands from cells that are not companion cells. In other words, the cell can limit the direction from which commands are accepted.

Feedback Port. Even if a cell outputs a pressure command once, the neighborhood is not always vacant. Therefore, the cell must output the pressure command several times. As such, we added one input-output port for feedback so that the cell can operate continuously with outputting a command to itself.

5 Explanation of Command Sets

We present three kinds of command sets that were obtained by trial and error. These command sets can increase an object to 1,024 (1, 2, 4, ..., 512, 1,024) cells. The correctness of the increase is checked by an ID that is distributed to objects.

5.1 Command Set 1

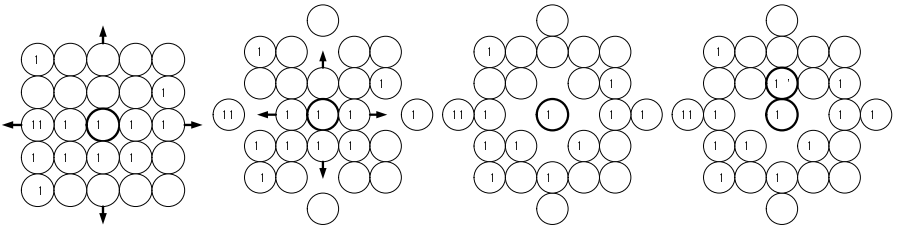
Command Set 1 indicates that an object that is going to duplicate itself places pressure only on the objects that exist in the vertical and horizontal directions, as shown in the figure. Therefore, it is not possible to increase the number of cells when the cells in the vertical and horizontal directions are all objects.

For example, consider the case of the cells shown in Fig. 4(a). Suppose Cell 13 wants to make a duplicate of itself. In this case, the pressure command is first sent in the vertical and horizontal directions. Then, Cells 3, 11, 15, and 23, which are at the outermost location in these direction are pushed out outward (Fig. 4(b)). Since locations next to Cell 13 are not vacant yet, the press command is resent, and Cells 8, 12, 14, and 18 are pushed outward, leaving a vacancy next to Cell 13 (Fig. 4(c)). Since the locations next to Cell 13 are now vacant, a clone can be generated (Fig. 4(d)).

5.2 Command Set 2

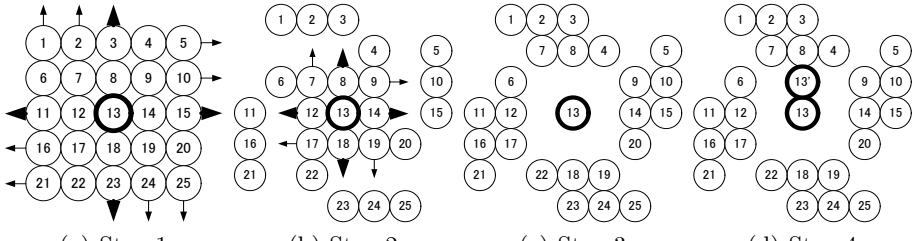
This command set has the ability to apply pressure in all directions.

In Fig. 5(a), when cell No. 13 increases, it sends "main pressure commands" to the surrounding cells (Nos. 8, 14, 18 and 12). Main pressure commands are

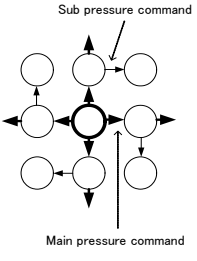


(a) Step 1 (b) Step 2 (c) Step 3 (d) Step 4

Fig. 4. Example operation in command set 1



(a) Step 1 (b) Step 2 (c) Step 3 (d) Step 4



(e) Pressure command

Fig. 5. Example operation in command set 2

transmitted directly and generate a "sub-pressure command" on the way (Fig. 5(e)). Cell Nos. 3, 15, 23 and 11, which received the main pressure command, are extruded. In addition, cell Nos. 1, 2, 5, 10, 25, 24, 16 and 21 are extruded by the sub-pressure command (Fig. 5(b)). Since the cells surrounding cell No. 13 are not vacant, cell No. 13 sends main pressure commands. As a result, cell Nos. 8, 14, 18 and 12) are extruded by this command, and cell Nos. 7, 9, 19 and 17) are extruded by sub-pressure commands (Fig. 5(c)). Since the cells surrounding cell No. 13 are vacant, a copy of cell No. 13 is made (No. 13') (Fig. 5(d)).

5.3 Command Set 3

This command set has a different method of applying pressure. A cell that is going to duplicate itself moves until a vacancy is found. The movement of the cell is realized by exchanging locations with the next cell until a vacancy is found.

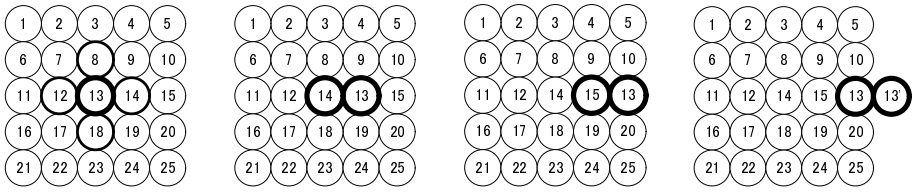


Fig. 6. Example operation in command set 3

In Fig. 6, Cell 13 gives increase demand to Cells 8, 12, 14, and 18 (Fig. 6(a)). Cell 13 exchanges locations with Cell 14, which gives the fastest acknowledgement (Fig. 6(b)). Cell 13 exchanges locations with Cell 15, in the previous direction (Fig. 6(c)). Cell 13 duplicates Cell 13' in the vacancy (Fig. 6(d)).

6 Evaluation of Command

6.1 Area of Implementation

In order to evaluate the mounting area, we compared the number of commands and the number of states among the command sets. The results are shown in Table 1.

This table shows the number of commands and the number of states for each command set. These values (number of states and number of command) are related to the mounting area. As a result, "command set 1" can be implemented in the smallest area and "command set 3" can be implemented in the largest area.

6.2 Rate of Effective Utilization

In order to evaluate the rate of effective use, we measured the rate of use at the time of increase until it became impossible for the number of cells to increase. The experiment was conducted for 12×12 cells. This is the number of cells of PCA-2 [10]. The rate of utilization was deduced from the number of objects when it becomes impossible to increase. A rate of utilization of 100% indicates that all of the cells can be used. The increase command is given randomly. The rate of utilization was obtained by averaging ten measurements.

The results are shown in Table 2. The rate of utilization for "command set 2" is 100%, indicating that all cells can be used ($12 \times 12 = 144$). Among the three

Table 1. Number of commands and number of states

Command set	Number of commands	Number of states
Command set 1	6	5
Command set 2	8	5
Command set 3	11	4

Table 2. Rate of utilization

Command set	Rate of utilization(%)
Command set 1	87.7
Command set 2	100.0
Command set 3	97.4

command sets, "command set 1" had the lowest rate of use of 87.7%. These results indicate that "command set 2" can use cells efficiently.

6.3 Result

The obtained results indicate that among the command sets, "command set 2" has the best performance. Since "command set 2" can use cells efficiently, it can be implemented in small areas.

7 Conclusion

In this paper, we proposed a new method of area management based on the concept of "pressure". In order to realize this method, we presented three command sets. Evaluation of the characteristics of these command sets revealed that "command set 2" gave the best performance. In this paper, the basic operation for a completely new area management method that requires no administrator was clarified.

References

1. K. Oguri, N. Imlig, H. Ito, K.Nagami, R. Konishi and T.Shiozawa, "General purpose computer architecture based on fully programmable logic," Proc. International Conf.Evolvable System (ICES'98), pp.323-334, 1998.
2. K.Nagami, K. OGURI, T. Shiozawa, H. Ito, and R.Konishi, "Plastic cell architecture: A scalable device architecture for general-purpose reconfigurable computing," IEICE Trans.Electron., vol.E81-C, no.9, pp1431-1437, Sept.1998.
3. Kiyoshi OGURI, Yuichiro SHIBATA, Nobuyoshi MURAKAMI, Akira KAWANO and Akira NAGOYA, "Asynchronous Bit-Serial Datapath for Object-oriented Rconfigurable Architecture PCA", IEICE TRANS.INF.&SYST., VOL.E82, No.1, JANUARY 2005
4. J.M.P.Cardoso and M.P. Vestias, "Architectures and Compilers to Support Reconfigurable Computing", ACM Crossroads, Spring 1999.
5. Hideharu AMANO, Yoshinori ADACHI, Satoshi TSUTSUMI and Kenfichiro ISHIKAWA, "A context dependent clock control mechanism for dynamically reconfigurable processors", IPSJ SIG Technical Reports, vol.2005 No.8, pp.13-22, 2005
6. Takayuki SUGAWARA, Keisuke IDE, Tomoyoshi SATO, "Dynamically Reconfigurable Processor Implementated with IPFlex's DAPDNA Technology," IEICE Transaction, vol.E87-D, no.8, pp.2004-2010, August 2004.

7. R. Konishi, H. Ito, H. Nakada, A. Nagoya, K. OGURI, N. Imlig, T. Shiozawa, M. Inamori, and K. Nagami, "PCA-1: A fully asynchronous, self-reconfigurable LSI," Proc. 7th International Symposium on Asynchronous Circuit and Systems (ASYNC 2001),pp.54-61, March 2001.
8. H. Ito, R. Konishi, H. Nakada, A. Nagoya, K. Oguri, A. Nagoya, N. Imling, K. Nagami, T. Shiozawa, and M. Inamori, "Dynamically reconfigurable logic LSI-PCA-1,"Proc.2001 Symposium on VLSI Circuits,pp103-106,June 2001.
9. M. Inamori, H. Nakada, R.Konishi, A. Nagoya, and K. Oguri, "A method of mapping finite state machine into PCA plastic parts," IEICE Trans.Fundamentals, vol.E00-A,no4,pp.804-810,April 2002.
10. H. Ito, R. Konishi, H. Nakada, H. Tsuboi and A. Nagoya, "Dynamically Reconfigurable Logic LSI designed as Fully Asynchronous System - PCA-2" , Proc. of COOL Chips VI, pp. 84, Apr. 2003
11. Tomoki KAMIYAMA, Keigo KURATA, Yousuke IKEHATA, Junji KITAMICHI and Kenichi KURODA, "Proposal and Implementation of Framework for Self-Reproductive applications on Dynamically Reconfigurable Device PCA", IPSJ SIG Technical Reports, vol.2005 No.8, pp.141- 146, 2005

Evaluation of Space Allocation Circuits

Shinya Kyusaka[1], Hayato Higuchi[1], Taichi Nagamoto[1],
Yuichiro Shibata[2], and Kiyoshi Oguri[2]

[1] Department of Electrical Engineering and Computer Science,
Graduate School of Science and Technology, Nagasaki University,
1–14 Bunkyo-machi, Nagasaki 852-8521, Japan
{qsaka, higuchi, taichi}@pca.cis.nagasaki-u.ac.jp
[2] Department of Electrical Engineering and Computer Science,
Nagasaki University, 1–14 Bunkyo-machi, Nagasaki 852-8521, Japan
{shibata, oguri}@cis.nagasaki-u.ac.jp

Abstract. This paper describes the design and evaluation of the PCA (Plastic Cell Architecture) cell, which implements a novel space allocation method. PCA is a dynamically reconfigurable architecture which exceeds the FPGA (Field Programmable Gate Array) in flexibility and generality. Circuit dynamically reconfiguration is achieved as administrators manage the heap areas. But, because objects operate and require new space in parallel, it is difficult to manage them collectively. So, we introduced the concept of pressure, which enables space allocation. As a simulation result, we found that this new method, which relies on pressure commands, could solve the problems of object management efficiently. We designed the PCA cell with space allocation capability. Consequently, the number of gates per PCA cell is 200, and the maximum delay time per cell is 3.55 ns. Moreover, the 3×3 PCA cell processing of six space-allocation commands consumes 306.3μW.

1 Introduction

Real operations of functions are performed by hardware. To perform arbitrary function, the hardware itself must have a variable part. However, if the hardware is composed of only variable parts, it cannot be configured or given any logic. Therefore, hardware is composed of dual components – a variable part and a fixed part.

Based on this consideration, we conducted a study on a novel architecture called PCA (Plastic Cell Architecture), which is different from conventional computers in its realization of the double dual-structure. In the structure of the PCA, SRAM-type FPGAs (Field Programmable Gate Arrays) are scattered over fine-grain networks spanning all of the chip. By making the best use of this structure, it can directly perform computing, *e.g.*, memory storage, processing and information transfer in parallel. In addition, the structure and scale of the configured circuits can be adapted to the behavior of applications by transferring the configuration information of the FPGAs. A unit of configuration of circuits is called an object in the PCA. Circuits are configured separately for each object

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 428–437, 2005.
© IFIP International Federation for Information Processing 2005

and are added into the system on-the-fly. The objects in the system that become unused are stopped and detached. To make adding and removing objects on-the-fly easy, PCA has taken the approach of using asynchronous circuits, which have been widely recognized as a means of designing low power consumption circuits and alleviating the performance degradation caused by its global clock skew.

The most difficult problem has been space allocation in the PCA. Because objects move in parallel, it is hard to manage them collectively. Thus, the *pressure command* was proposed as a new space allocation method to solve the abovementioned problem. First, we created a simulator to evaluate the pressure concept, and then we developed three pressure command sets. Next, we decided to implement one command set and evaluate its performance. This evaluation is presented here.

The composition of this paper is as follows. In Section 2, a description of PCAs along with related work is presented as the background of this study. In Section 3, details of the mechanism of the pressure command are described. In Section 4, the pressure commands that we designed are described. In Section 5, the evaluation is described. The paper ends with a short summary in Section 6.

2 Background

2.1 Plastic Cell Architecture

PCA (Plastic Cell Architecture) [1]–[3] is a dynamically reconfigurable architecture enhanced by the flexibility and the generality of the FPGA. PCA-1, developed by NTT (Nippon Telegraph and Telephone Corporation), was the first LSI to realize this concept. This LSI is an array of PCA cells, each of which

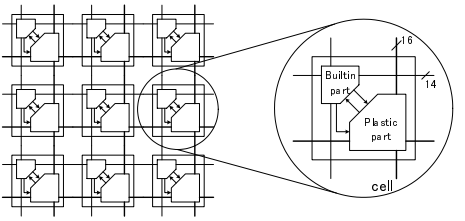


Fig. 1. Plastic Cell Architecture

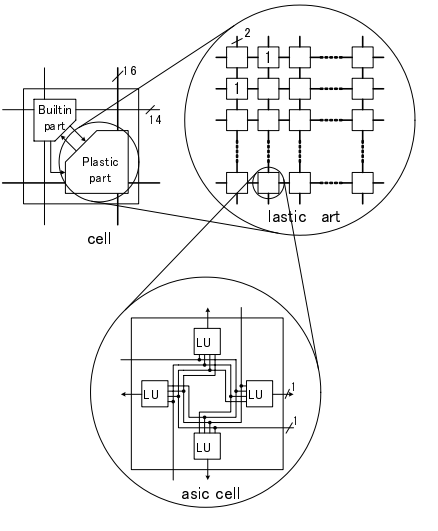


Fig. 2. Structure of plastic part

is composed of a plastic part and a built-in part (Fig. 1). The PCA cells are expected to realize the high integration of the LSI because of its homogeneous structure. Each PCA cell is connected with the adjoining cells in four directions. The plastic part works as both reconfigurable logic and local memory.

When it operates as a circuit, a circuit of arbitrary size is composed by connecting the plastic parts. When it operates as memory, the plastic part in the PCA cell is used with the unit. On the other hand, the built-in part connects the circuit and the memory in the plastic part and performs the data communication. The plastic part is composed of 8×8 two-dimensional basic cell mesh (Fig. 2). The basic cell is connected to four adjoining basic cells with 1-bit input and 1-bit output. Moreover, it has a 64-bit memory unit because it is composed of four 16-bit LUTs (look-up tables). This structure of the plastic cell that is composed of LUTs is called the Sea of LUTs.

The bit-serial PCA is an advanced PCA that adopts the asynchronous circuit system to a bit-serial data path. Asynchronous designs are clockless, so clock skews are avoided and an object can be designed with completely independent timing constraints. PCA operates by objects and messages, and the circuit structure based on local handshaking is also suited for miniaturization because it dispenses with long lines. A processing function is realized by state machines and shift registers in Bit Serial PCA. This coarse-grained composition can acquire efficiency higher than the Sea of LUTs.

2.2 Related Work

PCA-2 is being developed by NTT [4]. PCA-1 uses a $0.35-\mu$m process, but PCA-2 uses a $0.14-\mu$m process. The number of basic cells of the plastic parts are enhanced to 256, so the number of LUTs in a PCA cell are 1024. The width of the passing of data between built-in parts is enhanced to 9 bits. And the circuit composition for throughputs, such as making the detailed pipelines between modules, is changed. With such improvements, the performance has become better in the degree of integration and operation speed. PCA-Chip2, an experimental device of the PCA that uses a synchronous circuit, was developed at Kyoto University. The granularity that minimizes the amount of resources for logical synthesis was found by the method of synthesizing the benchmark circuit of MCNC to the various granularities [5]–[7].

The DRP (Dynamically Reconfigurable Processor) of NEC Electronics Corp. is a coarse-grained multi-context device [8]. The prototype chip DRP-1 was developed as the first LSI to use DRP architecture. DRP-1 has DRP cores, memory controllers, PCI controllers, and multipliers. The processing element (PE) of the DRP that is the reconfigurable element is the 8-bit arithmetic and logic unit/data management unit (ALU/DMU). The Digital Application Processor with Distributed Network Architecture (DAP/DNA), which IPFlex announced in 2002 [9], consists of parallel execution engine DNA matrices which are mounted for the exclusive use of the 32-bit RISC and 144 data processing elements. The DAP RISC core controls the processor's dynamic reconfiguration, while portions of an application that require high-speed data processing are han-

dled by the PE Matrix, which provides flexible parallel and pipelined operation. A DNA configuration memory consists of four banks. The Xpp of PACT XPP Technologies, Inc [10] consists of 128 32-bit processing array elements (PAEs). The PAEs and corresponding I/Os are segmented into two processing array cluster (PAC) blocks. A supervising confirmation manager (SCM) governs the configuration handling to local configuration managers, which further provide the necessary connectivity and processing for an algorithm.

3 Pressure Command

3.1 Space Allocation

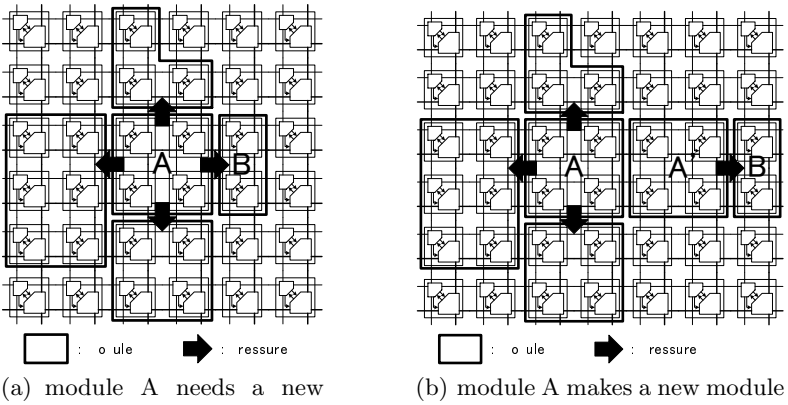
The domain administrator of a conventional PCA manages space allocation. It receives the demand of a new domain from some objects, and pays out some new domains. But it has the following problems.

–An administrator has to always know all objects and vacant areas, because these consume unnecessary electric power. This ruins the advantage of an asynchronous circuit.

–It is a bottleneck of processing that a lot of requests concentrate on the administrator. This disrupts the essential parallel processing of the PCA.

–The method of reducing overhead by dividing the administrator has been proposed, but this method is thought to cause various problems because communication between the administrators is difficult.

To solve the abovementioned problem, the *proliferation protocol* with the *pressure command* was proposed as a new method of space allocation. The concept is shown in Fig. 3. The communication among modules is performed through the built-in parts. Space allocation is also performed through the built-in parts with the proliferation protocol. If the module **A** needs the new module **A'**, it sends the pressure command to the surrounding modules, as shown in Fig. 3 (a).



(a) module A needs a new module

(b) module A makes a new module A'

Fig. 3. Pressure command

0 EMPTY

input-cmd	next-state	command	direction	lock
ISEMPTY	0	FIX	BACK	unlock
PRESS1	4	ISEMPTY	BACK	BACK
PRESS2	4	ISEMPTY	BACK	BACK
ERASE	0	–	–	unlock
MAKE	0	FIX	BACK	unlock
INCRZ	0	–	–	unlock
FIX	0	–	–	unlock
START	1	–	–	unlock

(a) EMPTY

1 ISMOD

input-cmd	next-state	command	direction	lock
ISEMPTY	2	MAKE	BACK	BACK
PRESS1	1	PRESS1	ST	unlock
PRESS2	1	PRESS2 PRESS1	ST RIGHT	unlock
ERASE	1	–	–	unlock
MAKE	1	FIX	BACK	unlock
INCRZ	1	–	–	unlock
FIX	1	–	–	unlock
START	3	INCRZ	RETURN	unlock

(b) ISMOD

2 DTSRC

input-cmd	next-state	command	direction	lock
ISEMPTY	1	FIX	BACK	unlock
PRESS1	2	–	–	remain
PRESS2	2	–	–	remain
ERASE	0	–	–	unlock
MAKE	1	FIX	BACK	unlock
INCRZ	2	–	–	remain
FIX	1	–	–	unlock
START	2	–	–	unlock

(c) DTSRC

3 TARGET

input-cmd	next-state	command	direction	lock
ISEMPTY	3	MAKE	BACK	BACK
PRESS1	3	PRESS1	ST	remain
PRESS2	3	PRESS2	ST	remain
ERASE	1	–	–	unlock
MAKE	3	FIX	BACK	remain
INCRZ	3	INCRZ PRESS2	RETURN ALL	remain
FIX	3	INCRZ	RETURN	unlock
START	3	–	–	remain

(d) TARGET

4 DTDST

input-cmd	next-state	command	direction	lock
ISEMPTY	0	FIX	BACK	unlock
PRESS1	4	–	–	remain
PRESS2	4	–	–	remain
ERASE	0	–	–	unlock
MAKE	1	ERASE	BACK	unlock
INCRZ	4	–	–	remain
FIX	0	–	–	unlock
START	4	–	–	unlock

(e) DTDST

Fig. 4. State machine table

Then, the pressure command spreads to each module. The surrounding modules which accepted this command move to a vacant place. In Fig. 3 (b), the module **B** moved and the module **A'** is made in the vacated space. In this method, space allocation can be performed without an administrator.

3.2 Basic Proliferation Protocol

Although some methods of realizing this proliferation protocol are considered, the following is considered the basic protocol. In order to simplify problems, we decided that routing lines between objects are not considered and an object consists of one module (= one PCA cell). The basic proliferation protocol operates with five states and eight Pressure commands. The state of "EMPTY" means a module is not a meaningful module. The state of "ISMOD" means a meaningful module. The "DTSRC" module sends its modular data. The "TARGET" is a proliferation module. This module needs a new domain. The "DTDST" module receives modular data.

Each state of a module reacts reasonably to the received command, as shown in Fig. 4. If a module receives a command (**input-cmd**), the module sends a **command** in some **direction**s. *ST* means straight. *RETURN* means the module sends a command to itself. A module may send out two or more commands (shaded areas). And then, its state changes to *next-state*. The term **lock** has three conditions. *unlock* means that the module can receive commands from all directions. *BACK* means that the module can receive from only the one direction which received the command immediately before. Also, lock conditions may *remain*. How these commands are used is explained in the following section.

3.3 Example of Proliferation Protocol

As an example, the situation that displays the module (from module-1 to module-25) in the state 'ISMOD' to the procession of 5x5 is shown in Fig. 5. The procession outside closes with the module of state 'EMPTY'.

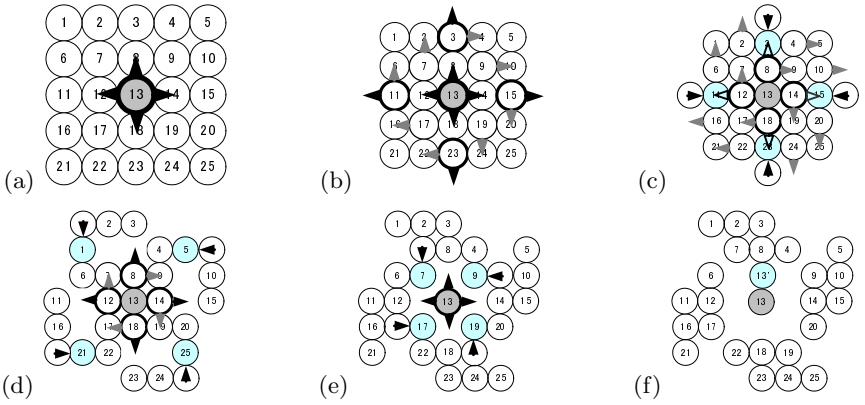


Fig. 5. Proliferation Protocol

To attempt module-13's module cloning, the command START is given. When a module accepts the command START, the module's state changes to 'TARGET' and sends itself the command INCRZ, as in Fig. 5 (a). Sending the command INCRZ is continued until the proliferation operation is finished. If a module accepts this INCRZ command, it sends the pressure command to neighbor modules (Fig. 5 (b)). The module which receives the command PRESS2 (►) sends out the same command in the same direction. However, in addition to the same direction, the module which received the command PRESS1 (►) sends out a command PRESS2 to the right. The pressure spreads by transmitting this operation through the modules. When the module of state 'EMPTY' receives the pressure, this module sends out the command ISEMPTY in the opposite direction (Fig. 5 (c)) and its state changes to 'DTDST'. This means it has become the destination of modular data. The module which receives command the ISEMPTY changes its state to 'DTSRC' in order to move data to itself. At this time, that module ignores any pressure commands. The module of state 'DTSRC' sends out the MAKE command to an empty module, and passes the modular data. The module which receives the data with the command MAKE returns the command ERASE. Then, the original module changes to the state 'EMPTY', and is not a module. It is shown in Fig. 5 (d). As a result of the repetition of the movement of such a module, the module of state 'EMPTY' is made near to the module of state 'TARGET'. Even if several empty modules are in the neighborhood (Fig. 5 (e)), the 'TARGET' module chooses only one module as a candidate for a copy. Finally, the module of the target makes its own clone in an empty module (Fig. 5 (f)).

In this simple protocol, pressure acts on the meaningful modules and gravitation acts on the module which is not a module.

4 Design of PCA Cell

This section describes the detailed inside specification and operation of the designed PCA cell which implements the pressure command.

4.1 Outline

We designed the PCA cell which implements the pressure command to evaluate the performance. Because this design focused on the justification and performance of the pressure command, the synchronous circuit system is adopted for design simplification.

4.2 Tools

SFL (Structured Function description Language), which is HDL (Hardware Description Language) developed at the NTT Research Institute for the logic composition tool PARTHENON (Parallel Architecture Refiner Theorized by NTT Original Concept), was used as HDL. SFL can make descriptions more elegantly than other languages can because it extracts the candidate for description to a single-layer synchronous circuit. 2SECONDS and PARTHENON were used as the simulator and logic composition tool for SFL, respectively. SFL2Verilog, which is one of the functions of PARTHENON, was used as a tool to change SFL into Verilog. and Design Compiler by Synopsis was used as the logic composition tool for Verilog.

4.3 Specification of PCA Cell

The PCA cell is shown in Fig. 6. The concrete functions of each part are described in the following.

–Input and output

Because the number of pressure commands is eight, as mentioned in the previous section, the input and the output need only 3 bits to express them. Moreover, there is a 3-bit register for memorizing the input command at the input ports.

–"*lock*" and "*looked*"

Because one *State Machine* can receive only one command at once, there are two 4-bit registers called *lock* and *looked* inside. These are used to control the commands.

lock is used to carry out the command processing exclusively. Each bit corresponds as follows. Namely, bit 1 (MSB) is south, bit 2 is west, bit 3 is east and bit 4 is north. Data processing can be performed where the bit is 1. (*e.g.*, If the value of *lock* is 0b1000, it means that the state machine receives only the command from the north.) The initial value of *lock* is 0b1111.

looked is used in order to memorize the direction of a command entering and going out. Each bit corresponds as in *lock*. If the corresponding value is 1 when a command enters, it shows that the command enters from that direction. If the corresponding value is 1 when a command goes out, it shows that the command goes out to that direction. (*e.g.*, If the value of *looked* is 0b0010 when a command enters, it means that the command enters from the west.) The initial value of *looked* is 0b0000.

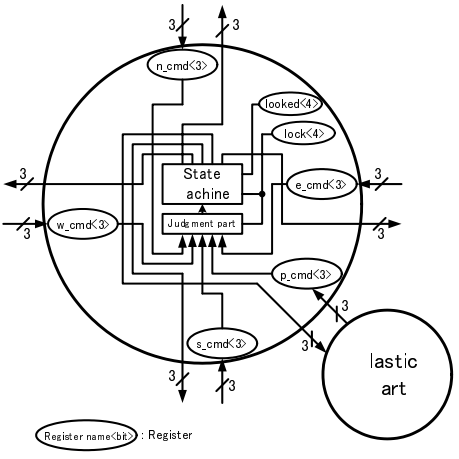


Fig. 6. PCA cell which implements the pressure command

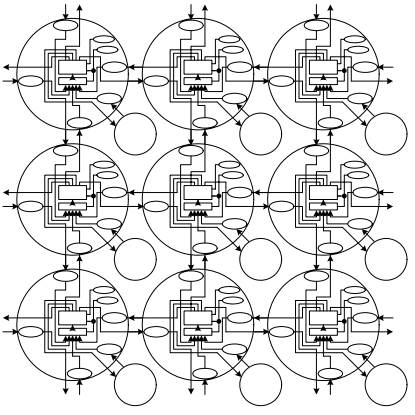


Fig. 7. 3×3 PCA cell

–*Judgment Part*

Input commands are sent to the portion called the *Judgment Part*. Here, the flag and *lock* of each direction investigate the commands before they enter the *Judgment Part*. If these bits are 1, the command from that direction can enter the *State Machine*. Moreover, as soon as the command goes into the *State Machine*, the bit of *looked* corresponding to the direction containing the command is set to 1, and the direction where the command was input is stored. If the following command doesn't enter from the direction that the command enters, the flag in that direction is adjusted to 0.

–*State Machine*

The *State Machine* receives input commands and determines the next command output and its direction. It can receive only one command at once. The *State Machine* has five states, and its processing is different from the command and the state described in the previous section.

5 Evaluation

We evaluated following items.

- Number of gates and maximum delay time of one PCA cell.
- Power consumption in an asynchronous circuit system of a 3×3 PCA cell (Fig. 7) processing six "START" commands.

We had to measure two power consumptions of the 3×3 PCA cell, i.e., processing six "START" commands and processing no commands, to determine the power consumption in the asynchronous circuit system. The power consumption of the 3×3 PCA cell processing six "START" commands consists of the power to process the actual commands and the power to make a clock signal. And

Table 1. Evaluation of one PCA cell

Number of gates of one PCA cell	200
Maximum delay time of one PCA cell	3.55 ns

Table 2. Evaluation of 3×3 PCA cell

Power consumption of 3×3 *PCA cell processing 6 commands*	16.0182 mW
Power consumption of 3×3 *PCA cell processing no commands*	15.7119 mW
Power consumption of 3×3 *PCA cell in an asynchronous circuit*	0.3063 mW(306.3 μW)

the power consumption of the 3×3 PCA cell processing no commands contains only the power to make a clock signal. Therefore, the result, which subtracts the latter from former, is the power consumption of the 3×3 PCA cell in an asynchronous circuit system which has no clock signals.

The number of gates and the maximum delay time of one PCA cell is shown in Table 1. Since the number of gates of the built-in part in PCA-1 is about 2500, the ratio of the space allocation part is about 7%. The maximum delay time of the space allocation is lightly affected because its value is 25 ns in PCA-1.

The power consumption of the 3×3 PCA cell processing the six "START" commands and processing nothing are shown Table 2. From these results, we could find the power consumption of the 3×3 PCA cell in an asynchronous circuit system. So, the power consumption of the 3×3 PCA cell per processing command is 51.0 μW.

6 Conclusion

This paper introduced command sets for the PCA. These command sets, based on the concept of pressure, were invented in order to realize a new space allocation method. After that, we implemented one command set in the PCA cell, and confirmed that it operated according to the protocol. The results of evaluating the performance of one PCA cell showed that the number of gates was 200, and the maximum delay time was 3.55 ns. Furthermore, the power consumption of the 3×3 PCA cell processing six commands in an asynchronous circuit system was 306.3 μW. Therefore, we found that the power consumption of the 3×3 PCA cell per processing command was 51.0μW. The ratio of the number of PCA cell gates (200) to the number of PCA gates of the built-in part (2500) showed that the space allocation part was only 7%. This fact indicates that it is possible to implement the space allocation part in the built-in part without large overhead.

Acknowledgment

The authors would like to express their sincere gratitude to the colleagues in their laboratory. The VLSI chip in this study was designed with PARTHENON,

Synopsys Design Compiler, Cadence and Avant! CAD tools through the chip fabrication program of the VLSI Design and Education Center (VDEC) at the University of Tokyo in collaboration with the Rohm Corporation and Toppan Printing Corporation.

References

1. K. Oguri, "New general-purpose information processing architecture PCA1 by cloth line theory", bit, January 2000, Vol. 32, No. 1, pp. 27-35 Kyoritsu Syuppan (2000).
2. K. Oguri, "New general-purpose information processing architecture PCA1 by cloth line theory", bit, March 2000, Vol. 32, No. 3, pp. 54-62 Kyoritsu Syuppan (2000).
3. K. Oguri, "New general-purpose information processing architecture PCA1 by cloth line theory", bit, July 2000, Vol. 32, No. 7, pp. 51-59 Kyoritsu Syuppan (2000).
4. H. Ito, R. Konishi, H. Nakata, A.Nagoya, "Dynamically reconfigurable method LSI PCA-2", The 1st reconfigurable system society, pp. 119-126 (2003).
5. K. Soda, "Design of LUT array type PLD and method of burying logical function", Kyoto university graduate school graduation thesis, (2000).
6. Y. Soga, N. Sugimoto, T. Izumi, T. Ogata, Y. nakamura, "Examination of granularity of PCA-chip2 based on logical mounting efficiency", The 19th PARTHENON society", pp. 15-20 (1999).
7. N. Sugimoto, A. Tomita, H. TsuTsumi, K. Sakai, K. Soda, Y. Nakamura, "Design and making for trial purposes of LUT array type PLD", VDEC LSI design forum, September (2000).
8. M. Motomura, "A Dynamically Reconfigurable Processor Architecture", Microprocessor Forum,October (2002).
9. http://www.ipflex.com
10. http://www.pactcorp.com

Automatic Configuration with *Conflets*

Justinian Oprescu, Franck Rousseau, and Andrzej Duda

LSR-IMAG Laboratory,
Grenoble – France
{Justinian.Oprescu, Franck.Rousseau, Andrzej.Duda}@imag.fr
www-lsr.imag.fr

Abstract. In this paper, we introduce a framework for the automatic configuration of applications running in dynamic environments where changes are frequent. We propose a way to describe, for each application, its configuration policy, and the execution environment's factors that affect its behavior. On this basis, we can generate application-specific configuration tools, called *conflets*. The application's source code is not required. The conflet combines the information drawn from the execution environment with the knowledge of the configuration policy. It is therefore able to detect when and how the execution environment modifies its characteristics, and can automatically react by reconfiguring the application and thus adapting it to the dynamic environment.

1 Introduction

Starting an application can be as simple as clicking an icon, or typing a sequence of characters in a terminal window. However, most of the time, several other operations must be fulfilled. Indeed, the applications require various configuration operations, such as provision of resources, editing of configuration files, or deployment of dependencies.

Under normal conditions, users can deal with configuration. Quite often, they are as well assisted by configuration assistants or wizards included by most of the modern operating systems or large applications. These assistants hide the complexity of the configuration, but are unaware of the dynamic nature of the execution environment. Their weakness lies in the interactive design, which supposes that the user can always initiate the configuration, and provide whatever information is needed. While not an issue in traditional computing environments evolving at human-speed, this missing feature becomes important in dynamic environments where changes are frequent, and potential interaction windows between applications are small. Hermann et al. evoke several mobile scenarios showing that, in dynamic environments, one of the most important resources is time [1]. Tennenhouse shares the same opinion, and predicts that the growing number of intelligent entities operating at faster-than-human speed pushes the human-in-the-loop computing to its limits [2]. More dynamic the environment is, less time remains between two successive modifications. Below a threshold of order of tens of seconds, it is clear that users cannot interfere anymore, as the rate of

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 438–447, 2005.
© IFIP International Federation for Information Processing 2005

modifications is too important. Any configuration activity going on for too long might be obsoleted before ending by the next occurring change. Kebhart and Chess see self-configuration as the first step toward autonomic-computing [3]. They explain that once the systems are able to configure automatically, we may look forward to self-optimization, self-healing, and self-protection as next milestones on the road to building systems that can manage themselves.

In this paper, we deal with automatic configuration for the applications running in dynamic environments. We introduce the concept of sequential configuration pattern that states that, for any application, the configuration can be divided into several sequential operations. We describe them in the Application Configuration Policy (ACP). During these operations, various interactions with the execution environment take place. We describe them in the Environment Awareness Policy (EAP). While the ACP focuses on the application's universe (like the existent configuration assistants), the EAP deals with the dynamism of the environment. We coined the name of *conflet* for configuration tools that implement both policies, being thus able to detect changes occurring in the environment, and to automatically react by adapting the application behavior. In this paper, we present the conflet architecture, discuss several design issues, and analyze the performances of our implementation.

2 Configuration Patterns

With respect to the application starting time, we identify three moments when configuration can be performed.

Before execution. Parameters that can be set-up before the application starts are usually persistent, typically stored on disk in configuration files.

At start-up time. Command-line parameters are provided by users or scripts.

At runtime. Runtime parameters are usually interactively provided to the application via a user interface.

All parameters are handled only during execution, and each one has a predetermined effect on the application, the order in which they are processed being less important. Usually, the persistent and the command-line parameters are processed only once, at start-up. Applications that allow only these two ways of configuration are said to follow a two-step configuration pattern. Applications that additionally allow runtime configuration follow a three-step configuration pattern. Both patterns are sequential, as shown on Figure 1, and we consider that most applications follow such a model.

Although a given parameter might be set-up in different ways, its effect is always the same. For instance, the URL of the document to be displayed in a web browser might be given in a persistent way as the default start-up page, but can be overridden by a URL provided on the command-line, or at runtime through the user interface. Even if the configuration time is different (before execution, at start-up, or at runtime), the effect of setting the parameter is the same: the browser loads and shows the document at the specified URL.

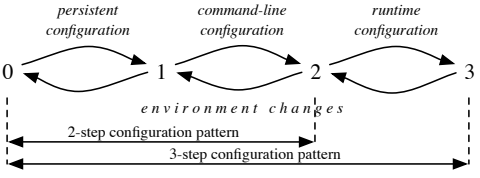


Fig. 1. Sequential configuration pattern

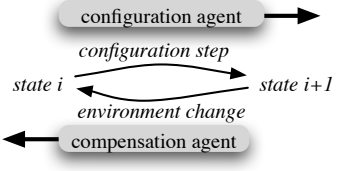


Fig. 2. Compensation agent

Nevertheless, the time at which the configuration of a specific parameter might take place is important, because it will affect the reconfiguration process. An application supporting only persistent and command-line parameters will have to be restarted for reconfiguration, whereas applications supporting runtime modification of their parameters won't need it, hence a shorter (re)configuration time, and fewer resources used. The sequential nature of the configuration patterns guarantees that minimizing configuration time amounts to minimizing the number of configuration steps as these operations are performed sequentially.

Configuration parameters provide information on the execution environment of an application, hence each configuration step leads the application closer to a state where its set-up is coherent with its environment. However, this set-up might get in an incoherent state after a change in the execution environment, thus requiring reconfiguration. The configuration state to which an application should return after an *environment change* is determined by the number of configuration steps needed to set up the modified parameters. For instance, a mailer for which the SMTP server can be given only through it's configuration file (persistent configuration) has to be restarted to cope with any environment change affecting this server, hence returning to state 0 of the process (see Figure 1). If runtime configuration had been an option, going back to state 2 would have been more efficient, avoiding restart.

Computing the shortest configuration sequence requires the knowledge of the parameters that have changed, i.e. needing reconfiguration, and the state in which their configuration is possible. To solve these issues we propose the Application Configuration Policy (ACP) and the Environment Awareness Policy (EAP), presented in the remainder of this section.

2.1 Application Configuration Policy

The Application Configuration Policy (ACP) describes the configuration operations. Each one is performed by a configuration agent that executes configuration commands, and uses a set of resources (parameters) provided by the execution environment.

Each configuration agent has a counterpart in the form of a compensation agent that can undo its actions, thus returning in a previous state (see Figure 2). The compensation agents have a roll-back effect. When the application is reconfigured, they bring it in the state from where the shortest configuration sequence (performed by configuration agents) can start.

The ACP specifies a list of parameters for which configuration agents need values as inputs. The Environment Awareness Policy defines how to get them.

2.2 Environment Awareness Policy

The Environment Awareness Policy (EAP) define how to gather up-to-date values for the configuration parameters. These parameters are resources provided by the execution environment and needed by the application, and whose modification should trigger reconfiguration, using the configuration agents specified by the ACP.

For each parameter, the EAP lists the appropriate tools that must be used to detect changes, and to fetch the most recent value. They are called sources, and for a given parameter, they are considered equivalent, as they implement different ways to obtain the same value, but have priorities defining the order in which they are inquired. For example, the parameter for an SMTP server might have three distinct sources, one relying on DHCP, the second using a service discovery protocol, the third asking input to the user.

By combining the ACP and EAP, we can detect when and how the execution environment modifies its characteristics, infer the state in which the application must return in order to be properly reconfigured, bring the application in that state by applying compensations, and finally perform the configuration. We built the configuration framework in order to address these issues.

3 Configuration Framework

The configuration framework is a generic piece of software that takes as entries an Application Configuration Policy (ACP) and the correspondent Environment Awareness Policy (EAP), and generates a conflet specific to the application. It was first developed as part of the OmniSphere project that aims at developing personalized communication spaces offering the user new applications built on-the-fly by data-flow composition from dynamically discovered services [4, 5, 6]. The conflet does not depend on the programming language of the application, and uses operating system standard facilities, such as process or file handling, implemented within configuration and compensation agents. We chose to implement the framework in Java, and to code both policies in XML.

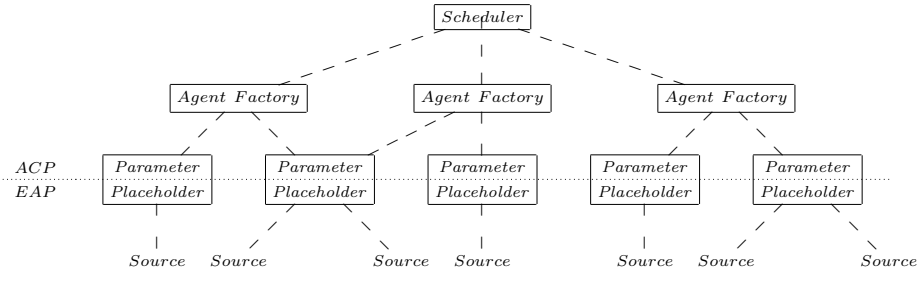


Fig. 3. Conflet architecture for a three-step configuration pattern

3.1 Conflet Architecture

A conflet has a tree-like structure. Figure 3 shows the conflet of an application that follows a three-step configuration pattern. The top element is the scheduler. It governs the execution of configuration and compensation agents. Agents are generated on-the-fly by agent factories whenever the value of one of subjacent parameters changes. Up-to-date parameter values are produced by parameter sources embedded in the environment.

The implementation of each component of the conflet depends on the application to be configured. However the basic behavior is provided through generic classes that can be specialized. The generic architecture follows the pipes-and-filters pattern, receiving event via an entry queue, performing its task, and notifying other registered components about the result of this task.

Parameter Placeholders. The sources send modifications events each time they detect a specific modification in the environment. The parameter placeholders listen for such events, analyze the conveyed modification, and decide whether to propagate it or not.

Low-Pass Filters. Allowing the components of the conflet to send notifications too often is an open door for denial-of-service attacks. Therefore, the parameter placeholders, and the agent factories implement output low-pass filters that prevent the components downstream from flooding or bursts. A filter behaves like a leaky bucket that can hold at most one item (or event). Although a typical leaky bucket would discard any item arriving when another is already in, the low-pass filter keeps the arriving item and discards the other. Each filter has a sending frequency. At each tick, the event in the bucket (which represents the last modification) is retrieved and transmitted to registered listeners. Ignoring several modifications that occur too quickly has no negative effect, as the agent factories and the scheduler (the components that receive modifications events) do not depend on the modification events history.

Agent Factories. The role of the agent factories is to gather up-to-date values from all managed parameters, and to generate agents. The factories receive modification events from parameters and, in response, generate new pairs of agents (configuration and compensation agents). An agent is initialized with the list of current parameter values, and a set, called Δ, of references to the parameters whose values have changed since the last agent passed through the factory low-pass filter.

Scheduler. Both new configuration and compensation agents are not executed immediately after creation. The exact moment of execution and their relative order is given by the conflet scheduler that has a global view of the application state. For example, a configuration agent that starts an application, cannot be executed if the application is already started. Before, the scheduler must rollback the current configuration by executing necessary compensation agents. In the next section, we discuss the scheduling algorithm that decides the agents to be executed and their order.

3.2 Scheduling Algorithm

To manage the application state, the scheduler uses a compensation stack. Every time a configuration agent is executed, the corresponding compensation agent is pushed into the stack. When the execution environment changes and obsoletes the current configuration, one or more compensation agents are retrieved from the stack and executed before the new configuration is applied. The last-in, first-out property of the stack guarantees that the compensations are performed in reverse order. The number of compensations that should be retrieved from the stack and executed depends on the new configuration and is calculated by the scheduling algorithm.

The algorithm uses a list containing the last generated agent for each agent factory. When a new agent is provided to the scheduler, it replaces, in the list, the previous agent generated by the same agent factory. As a new agent means that the application must be reconfigured, the scheduler must then decide what compensation and configuration agents should execute. It calculates therefore the shortest configuration sequence.

First, it computes the global Δ set for all agents, as the union of Δ sets of all agents in the list:

$$\Delta = \Delta_1 \cup \Delta_2 \cup \cdots \cup \Delta_n.$$

The global Δ set contains references to all parameters whose values have changed since the last configuration, and needs to be addressed by the current one. Executing all agents surely configures them all, and hence represents a valid method of reconfiguration. However, it implies as well that the longest possible sequence of configuration agents is executed. And because different agents can configure the same parameter (that is, a parameter can appear in the Δ set of several agents), there may be a shorter sequence with the same Δ set. The scheduling algorithm then calculates m, with the following properties:

$$\Delta = \bigcup_{i=m}^{n} \Delta_i,\ m \in \{1 \ldots n\},\ m \text{ is maximal}$$

Because it is maximal, m gives the minimal number of agents $(n - m + 1)$ that must be executed in order to reconfigure all parameters in the global Δ set. It also represents the number of compensation agents to be retrieved from the compensation stack and executed before. As configuration agents m to n are executed, their compensation agents are pushed into the compensation stack.

The scheduling algorithm calculates a sequence of agents, which reconfigures all parameters referenced in the global Δ set. The maximum property of m guarantees that the result is the shortest valid sequence. Additionally, because of the sequential nature of the configuration pattern, all the other valid sequences would be longer and would include the calculated sequence. Therefore, the scheduling algorithm indirectly minimizes the length and the resources used by the configuration.

3.3 XML Policies

The configuration framework generates an application conflet on the base of information provided by the ACP and EAP policies. Both are declarative and grouped in a single XML document. We have defined the following DTD.

All the components of a conflet (scheduler, agent factories, parameter placeholders, and parameter sources) are represented by Java classes, hence we need a serialization mechanism to marshal and unmarshal their state to and from XML. Currently we support Jox serialization[1].

```
<!ELEMENT conflet (EAP, ACP, state)>
<!ELEMENT state (serial?, ANY?)>
<!ATTLIST state
    class CDATA #REQUIRED  <!-- URL: Java class to be loaded. -->
>
<!ELEMENT serial (#PCDATA)> <!-- URL of an external file containing the
    object serialization. If empty, the serialization follows inline. --
    >
<!ATTLIST serial
    tool CDATA #REQUIRED  <!-- The serialization mechanism: "jox". -->
>
```

The XML element named state contains the serialization. The Java class is dynamically loaded, as its URL is provided as an attribute of the state element. The object is created using the no-arguments constructor, and can be further initialized using the serialization mechanism.

EAP. The Environment Awareness Policy defines the list of parameters and, for each one, a unique identifier and a list of sources. The state element contains the serialization of every Java object.

```
<!ELEMENT EAP (parameter)*>
<!ELEMENT parameter (source+, state)>
<!ATTLIST parameter
    ref ID #REQUIRED <!-- The unique parameter identifier. -->
>
<!ELEMENT source (state)>
```

ACP. The Application Configuration Policy contains the list of agent factories, and the agents scheduler. For each agent factory, a unique identifier and the list of parameter references are provided. Each reference uniquely identifies a previously defined parameter.

```
<!ELEMENT ACP (factory+, scheduler)>
<!ELEMENT factory (pref*, state)>
<!ATTLIST factory
    ref ID #REQUIRED <!-- The unique factory identifier. -->
>
<!ELEMENT pref EMPTY>
<!ATTLIST pref
    id IDREF #REQUIRED <!-- A reference to a parameter. -->
>
```

The scheduler manages the order in which the agents are executed. It is therefore given the sorted list of agent factories. The order is important as it is the order in which the generated agents will be executed.

[1] `http://www.wutka.com/jox.html`

```
<!ELEMENT scheduler (fref+,state)>
<!ELEMENT fref EMPTY>
<!ATTLIST fref
    id IDREF #REQUIRED <!-- A reference to an agent factory. -->
>
```

3.4 Performance Issues

We ran several experiments in order to validate our proposal: we measured (i) the time taken by the conflet to produce valid configuration sequences, (ii) the conflet generation time, and (iii) its memory consumption. We carried out all experiments on an Intel Pentium M 735 1.7 GHz laptop with 512 MB of memory running Windows XP and the Java Virtual Machine 1.5 as well as an Intel Pentium III 650 MHz laptop running Linux Fedora Core 3 and the JVM 1.5. All the results represent the average of at least 10 trials.

The measured time between the time a source detects an *environment change* and the time the configuration sequence is ready to be executed is insignificant. For conflets with less than 10 agent factories and with low-pass filters that do not introduce any delays (their frequency is greater than the rate of modifications), observed time was less than 1 ms.

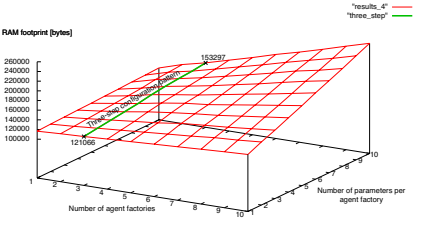


Fig. 4. RAM footprint of conflets

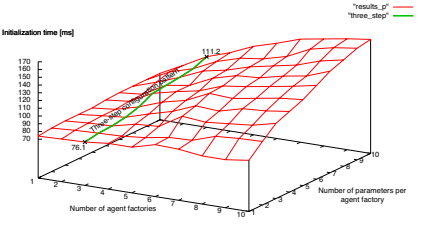


Fig. 5. Conflet generation time

The conflet RAM footprint varies with the number of agent factories, and with the number of parameters per factory, as it is shown on Figure 4. The footprint of a typical three-step conflet is between 121066 and 153297 bytes. Note that only the Java object heap was considered here, thus ignoring the memory occupied by Java code and thread stacks.

Figure 5 shows the conflet generation time (parsing the XML file, and creating all Java objects) according to the number of agent factories and the number of parameters per factory. Generation times are under 200 ms in all cases. However, generating huge conflets (50 agent factories with 50 parameters each) can take almost 3.5 seconds.

4 Related Work

Automatic configuration needs to embrace two important issues: systems need to detect changing environmental conditions or changing system capabilities and

to react appropriately. The ability to deal with the dynamism of the execution environment is the principal concern of service discovery protocols [7, 8, 9, 10]. Within our configuration framework such tools can be wrapped by parameter sources, and used to detect changes, such as services that are added, or removed from the network.

While service discovery protocols are considered well suited for ad-hoc environments, DHCP [11] is preferred in administered environments. DHCP allows clients to obtain configuration information from a local server manually initialized by an administrator. Akin to service discovery protocols, DHCP clients can be wrapped by parameter sources.

Self-configuration is addressed by several projects that, still, focus mainly on component-based applications. Fabry explained in 1976 how to develop a system in which modules can be changed on the fly [12]. Henceforth, lots of projects dealing with the modification of interconnections between components appeared [13, 14, 15]. Such approaches can be employed by configuration agents specific to component-based applications.

The Harmony project, proposed by Keleher et al. [16, 17], allows applications to export tuning alternatives to a higher-level system, by exposing different parameters that can be changed at runtime. Although Harmony and the conflet framework share the same vision, the former focuses on performance tuning and requires that applications are "Harmony-aware". Conversely, the conflet framework can configure applications that were not designed for automatic configuration and whose source code is not available.

5 Conclusion

In this paper, we presented a configuration framework that allows applications to be automatically (re)configured. Automatic configuration is made possible by separating the information about the configuration in two complementary classes. The first describes the application configuration policy, while the second details the external factors that influence the application. The conflet combines both: it monitors the execution environment and reacts to the modifications of external factors by reconfiguring the application accordingly. An external tool, the conflet, that is automatically generated from an XML description, manages the application's execution to adapt it to the dynamism of the execution environment.

A significant advantage of separating the configuration policies from the execution environment awareness, and externalizing them into XML, is the ability to add new detection tools without modifying the application.

References

1. Hermann, R., Husemann, D., Moser, M., Nidd, M., Rohner, C., Schade, A.: DEAPspace – Transient Ad Hoc Networking of Pervasive Devices. Computer Networks **35** (2001) 411–428
2. Tennenhouse, D.: Proactive Computing. Comm. of the ACM **43** (2000) 43–50

3. Kephart, J., Chess, D.: The Vision of Autonomic Computing. IEEE Computer Magazine **36** (2003) 41–50
4. Rousseau, F., Oprescu, J., Paun, L.S., Duda, A.: Omnisphere: a Personal Communication Environment. In: Proceedings of HICSS-36, Big Island, Hawaii (2003)
5. Oprescu, J., Rousseau, F., Paun, L.S., Duda, A.: Push Driven Service Composition in Personal Communication Environments. In: Proceedings of PWC 2003, Venice, Italy (2003)
6. Oprescu, J.: Service Discovery and Composition in Ambient Networks. PhD thesis, Institut National Polytechnique Grenoble (2004) in french.
7. Guttman, E., Perkins, C., Veizades, J., Day, M.: Service Location Protocol, Version 2. IETF RFC 2608, Network Working Group (1999)
8. Jini Community℠: Jini™ Architecture Specification (2005) `http://www.jini.org/standards`.
9. UPnP Forum: UPnP™ Device Architecture 1.0 (2003) Version 1.0.1 `http://www.upnp.org/resources/documents.asp`.
10. Cheshire, S., Krochmal, M.: DNS-Based Service Discovery. IETF draft (2004) Expires August 14, 2004.
11. Droms, R.: Dynamic Host Configuration Protocol. IETF RFC 2131, Network Working Group (1997)
12. Fabry, R.: How to design a system in which modules can be changed on the fly. In: 2nd Intl Conf. on Software Engineering. (1976)
13. Plasil, F., Balek, D., Janecek, R.: SOFA/DCUP: Architecture for Component Trading and Dynamic Updating. In: Proceedings of ICDCS 1998. (1998)
14. De Palma, N., Bellissard, L., Riveill, M.: Dynamic Reconfiguration of Agent-based Applications. In: The European Research Seminar on Advances in Distributed systems (ERSADS). (1999)
15. Batista, T., Rodriguez, N.: Dynamic Reconfiguration of Component-Based Applications. In: Intl Symp. on Software Engineering for Parallel and Distributed Systems. (2000)
16. Keleher, P., Hollingsworth, J.K., Perkovic, D.: Exploiting Application Alternatives. In: Proceedings of ICDCS 1999. (1999)
17. Ţăpuş, C., Chung, I.H., Hollingsworth, J.: Active Harmony: Towards Automated Performance Tuning. In: Proceedings of SuperComputing. (2002)

Path Concepts for a Reconfigurable Bit-Serial Synchronous Architecture*

Florian Dittmann[1], Achim Rettberg[2], and Raphael Weber[2]

[1] University Paderborn/HNI, Fuerstenallee 11, 33102 Paderborn, Germany
[2] University Paderborn/C-LAB, Fuerstenallee 11, 33102 Paderborn, Germany
{roichen, syne}@upb.de, achim@c-lab.de

Abstract. This paper develops path concepts for the execution of different algorithms on a reconfigurable architecture. New architecture concepts demand for permanent evaluation of such extensions, also including validating case studies. The recently patented synchronous bit-serial pipelined architecture, which we investigate in this paper, comprises synchronous and systematic bit-serial processing without a central controlling instance. It targets future high speed applications due to the abdication of long wires. The application specificity of the basic version of the architecture can be overcome by so called routers, achieving a reconfigurable system. This paper focuses on the difficulty to conceptualize these routers and proposes several variations for implementation. The case study, which comprises a combined version of the FDCT/IDCT algorithm, serves as an application example for the reconfigurability of the architecture. The example – implementing both algorithms in one operator network – broadens the application area of the architecture significantly.

1 Introduction

The bit-serial architecture (referred to as MACT – Mauro, Achim, Christophe and Tom), which we examine and extend in this paper, was invented in response to current problems of chip design. MACT combines ideas of asynchronous design and bit-serial processing, and represents a synchronous and pipelined architecture without central controlling instance.

Implementations of the so far presented version of the MACT architecture would be application specific [1, 2]. This limitation is only partly in line with current market conditions of the intended application area: data-stream oriented processing (e. g., image compression or filtering).

In this paper, we investigate how to realize multiple paths, and how to model and implement routers and multiplexors to enable path merging and path selection. We abstract and conceptualize graph variations capable for path selection. Strict categorization helps us to distinguish possible cases and to develop solutions for the high level synthesis of such elements. Thereby, we can fall back on

* This work was partly funded by the Deutsche Forschungsgemeinschaft (DFG) in SPP 1148.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 448–457, 2005.
© IFIP International Federation for Information Processing 2005

an already realized high level synthesis for the architecture [3, 4], which automatically generates MACT implementations in VHDL out of data flow graphs. In the high level synthesis, we avoid deadlocks and establish the local control mechanism of the MACT architecture.

As an example for the effectiveness of the extended MACT architecture, we implemented parts of the MPEG-2 algorithm with the components of our architecture.

The rest of this paper is organized as follows. First, we resume related work. Next, we shortly explain the MACT architecture, summarizing the main aspects and benefits of the architecture. Then, we systematically introduce concepts for multi paths within the MACT architecture, including detailed description of the extension. Section 5 familiarizes the reader with the example: combining IDCT and FDCT. Section 7 sums up with a conclusion and gives an outlook.

2 Related Work

Adding routers to the MACT architecture means extending MACT towards a reconfigurable architecture. In the literature, we find several concepts, which can be related to the reconfigurable MACT architecture and give some basic information for our router conceptualization.

The concept of wormhole run time reconfiguration [7] relies on a distributed control scheme and therefore avoids central controllers. Wormhole run time reconfiguration is based on the stream concept. The idea is implemented in the Colt Configurable Computing Machine [8]. Our architecture relies on a similar concept. In contrast to wormhole reconfiguration, we transport only the information when and which path to select and not the whole reconfiguration stream.

So-called self-reconfiguration [9] can be seen as one step towards easing the control process, as the part of the controller tracking the processing can become obsolete. Yet, advanced concepts are needed. One idea is to locate the state machine inside the FPGA, either in a soft- or hard-core processor. Considering Xilinx's Virtex FPGAs, both is possible using MicroBlaze or a PowerPC.

In [10], we find basic work for deadlock-free routing. The authors introduce virtual channels to achieve deadlock-freeness. In principle, MACT relies on a similar concept; yet, the deadlock-freeness is realized by the local control mechanism triggered by data packets traversing the architecture. Thus, the architecture itself bears the capability to avoid deadlocks. Deadlock avoidance is a critical issue during the high level synthesis especially if routers are added to the MACT architecture (see below).

3 MACT Architecture

MACT is an architecture, that breaks with classical design paradigms. Its development came in combination with a design paradigm shift to adapt to market requirements. The architecture is based on small and distributed local control units instead of a global control instance. The communication of the components is realized by a special handshake mechanism driven by the local control units.

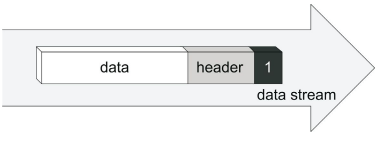


Fig. 1. Example data packet

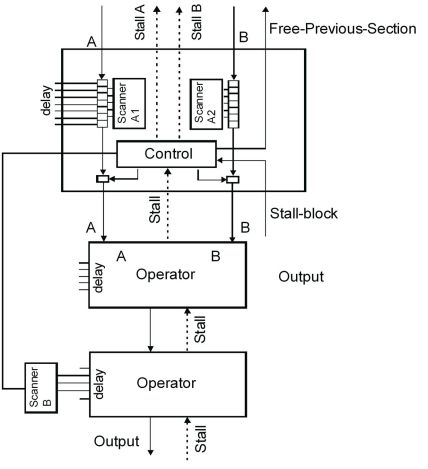


Fig. 2. Synchronisation

MACT is a synchronous, de-centralized and self-controlling architecture. Data and control information are combined into one packet and are shifted through a network of operators using one single wire only (refer to Figure 1).

The controlling operates locally only based on arriving data. Thus, there exist no long control wires, which would limit the operating speed due to possible wire delays. This is similar to several approaches of asynchronous architectures and enables high operation frequency. Yet, the architecture operates synchronous, thus enabling accurate estimation of latency, etc. a priori.

MACT operates bit-serial. Bit-serial operators are more area efficient than their parallel counterparts. The drawback of bit-serial processing is the increase in latency. Therefore, MACT uses pipelining, i. e., there exist no buffers, operators are placed following each other immediately.

Thus, MACT resembles a systematic bit-serial architecture. It enables the user to benefit from the advantages of bit-serial processing like low area consumption, significant reduction of I/O pins (serial instead of parallel communication), while offering a reliable pipelined system.

Further, processing increases to be more and more bit-serial. Systems nowadays increasingly use serial communication, while still processing in parallel. Using MACT enables integrated bit-serial processing, avoiding discontinuity concerning the bit-width, i. e., no parallel/serial conversion is needed.

Implementations of MACT are based on data flow graphs. The nodes of these graphs are directly connected, similar to a shift register. Synchronization of different path lengths at nodes with multiple input ports is resolved by a stall mechanism, i. e., the shorter paths, whose data arrives earlier, will be stalled until all data is available (refer to Figure 2). The necessary stall signals run in opposite to the processing direction and are locally limited, in order to avoid a complete stall in the preceding pipeline. The limitation is realized by a so called *block stall* signal, which is locally tapped in a well defined distance.

We consider the flow of data through the operator network as processing in waves, i. e., valid data alternates with gaps. Due to a sophisticated interlock mechanism adapted from asynchronous design, the gap will not fall below an individual lower bound. Thus, the MACT implements a fully interlocked pipeline. In combination with the developed high level synthesis, the MACT guarantees deadlock free processing. The corresponding signal is the so called *free previous section* signal, which is generated by small logic in each synchronizing control block (see Figure 2). These control blocks are found at each multiple input operator and synchronize packets arriving at different instances of time. The architecture is described in more detail in [1, 2].

4 Router Design

Specialized routing components are further extensions of the architecture to allow data-driven reconfiguration. Therefore, we interweave data and configuration information by processing packets including both. So far, the routers only have been conceptualized in [11]. In order to efficiently use routers in the MACT architecture, this section introduces the routing concept abstractly.

Routers are placed inside the data flow graph and allow for different processing paths. Each data packet carries routing information or information for unique identification, thus enabling path selection. The routers process the header and/or the data section of each data packet and forward the packet to the corresponding path. This feature allows space saving and flexible implementations of data flow graphs, as multiple algorithms or different characteristics of algorithms may be present in the same implementation of MACT. As example, we may select different compression granularity or increase processing accuracy using router based MACT. The re-use of common sections reduces the overall area requirement. In the following, we formalize graphs containing routers abstractly.

4.1 Variants of Multiple Path Folding into One Operator Network

There exist several variants how routers may be present in graphs, which we will categorize and explain below. In addition to routers, we have to integrate the corresponding counter part: multiplexors. Thus, the two operators, which enable multiple paths within one processing graph are routers (R) and multiplexors (M). Routers act like de-multiplexors.

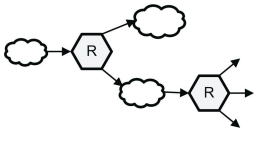


Fig. 3. Tree with one input and multiple output variations

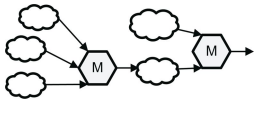


Fig. 4. Tree with variable inputs leading to one output

Tree The simple tree case comprises two alternatives as displayed in Figure 3 and Figure 4. In the first, we only find routers, which distribute incoming packets to the appropriate output, while in the latter, we combine multiple inputs to one output via multiplexors.

Fork – Join In Figure 5 a, we display the possibility of alternative paths. There, data packets arrive at the router and are forwarded to one path according to their routing information. This kind of path option can be reasonable for data stream processing, where only parts are different, e. g., encoding granularity varies.

Join – Fork In contrast to the latter characteristic, Figure 5 b shows how similar sections within a processing algorithm can be utilized using MACT with routers.

Combinations The above mentioned basic characteristics can be combined in several ways. Apart from a random combination, there may exist an ordered structure, e. g., mesh-based. While random based versions require extended generation algorithms, mesh-based structures can base on principles of systolic arrays. These concepts are out of the scope of this paper.

4.2 Path Encoding

We can encode the information for path selection into several parts of the data packet (refer to Figure 1). Thereby, we can indicate different paths using only a few bits due to logarithmic dependency.

Header We can use the header to hold the routing information, i. e., the information for the path selection. This localization is the most appropriate, as headers precede in the data packet and their content is more easily accessible.

Data If we make a path decision based on the content of the data, we introduce if-then-else to the MACT architecture. We can use this style, when specific paths are only needed, after the data has passed a threshold, etc.

Header + Data When we use both section of a data packet, we can process both by the routers. Thus, we can base the path selection not only on the header information, but also on the data content itself. Thus, individual routers forward packets depending on either the control information present in the header, the data content present in the data section, or on both.

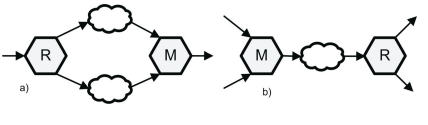


Fig. 5. a) Fork and Join, b) Join and Fork

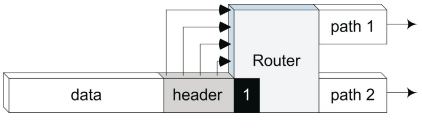


Fig. 6. Router implementation

4.3 Router Implementation

Figure 6 shows an exemplary implementation of a router. Routers tap the information of approaching data packets, when the preceding '1' of every packet arrives at the router. Within one step (clock cycle), the router processes the information and subsequently forwards the data packet to the appropriate path based on predefined table entries. Yet, the paradigms, which lie behind the routers, can follow different principles.

Basic In the basic version, the router only reads the information, selects the appropriate path for the packet and forwards the packet without modifications.

Consuming In a more advanced concept, the router can consume parts of the information of the header, i. e., the corresponding section in the header is removed. Thus, we can decrease the packet length, which can lead into less area needed and shorter wire lengths.

Additive Further, routers may extend packets by additional information. This possibility makes sense, e. g., when there exists a short common section, and packets must be distinguished directly after this section.

4.4 High Level Synthesis and Extensions

Both routers and multiplexors represent new parts of the MACT architecture and therefore must be considered by the high level synthesis. Thereby, we can mainly rely on our already developed high level synthesis.

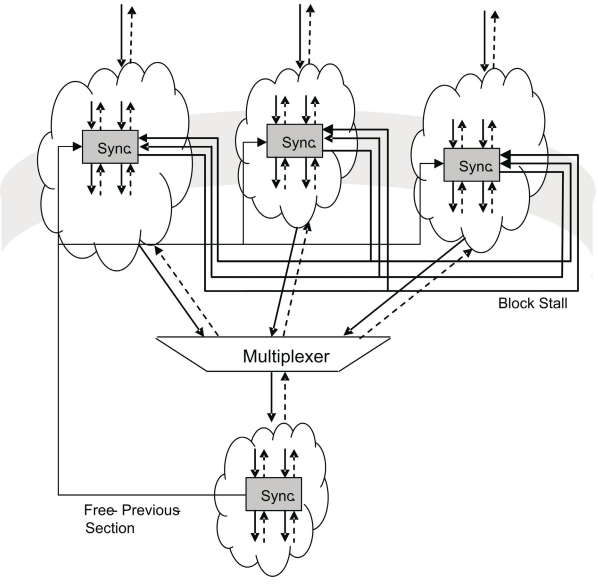


Fig. 7. Multiplexer realization

Router: As routers distribute packets to different paths only, and therefore do not affect preceding paths, they can be treated as normal operator elements by the high level synthesis. In detail, they accept new packets, when all possible succeeding paths are empty, i. e., no packet is present within the tapping range of the control signals.

Multiplexer: In contrast, multiplexer re-unit data paths, i. e., they combine more than one line. The arrival of two packets at the same or nearly same time can cause problems due to false free path information.

The deadlock free processing of the MACT architecture relies on gap and data periods (processing in waves). In order to guarantee a minimal lower bound between two data packets, we enhance multiplexors by *block stall* signals (refer to Figure 7). These signals are activated as soon as one data packet reaches the area between the last synchronizer and the multiplexer of the current path. All other similar areas of the other paths preceding the multiplexer receive a block stall signal, and thus deny arriving packets to enter this section. The packet of the one valid path can be processed without interferences.

Yet, all sections must be freed again, which is done by the *free previous section* signal, which is generated by local logic when the data packet has exited the critical section. It is received by all synchronizers preceding the multiplexer and frees all corresponding paths.

Up-to-date technologies of chip design enable to increase the router concept further. So far, all possible paths must be known at implementation time. Routers enable dynamic path selection between predefined processing paths. Reconfigurable devices (e. g. Xillinx Virtex FPGAs) possess the capability of partial run time reconfiguration. Thus, we can add additional paths of the MACT architecture including routers to our system during run-time (using partial reconfiguration), i. e., new path alternatives can be realized.

We thus achieve a specific self-controlled run time reconfiguration. Generally speaking, during run time reconfiguration, tasks are dynamically loaded into the reconfigurable processing unit on demand. The area currently reconfigured does not influence other regions in operation at the same time. In the MACT version, we additionally avoid a usually large and complex central control entity. There is no need to track the data in the operator network by a central control unit, in order to activate reconfiguration. The request for reconfiguration is generated locally.

Therefore, we modify the routers of MACT. Upon arrival of a new packet, the modified router detects the information needed (header and/or data section) and checks for the availability of the required data path. If the needed path is not resident, the router activates the run time reconfiguration.

5 Case Study

We use as a case study some tasks of MPEG-2. These are the FDCT/IDCT and some video conversion formats. The FDCT/IDCT algorithm is implemented according to the Chen/Wang [5,6] approach. The task of the FDCT is to transform

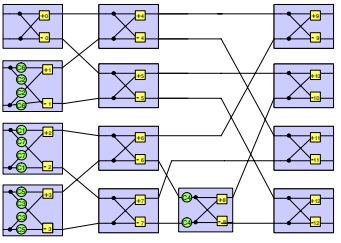


Fig. 8. Inverse Discrete Cosine Transformation

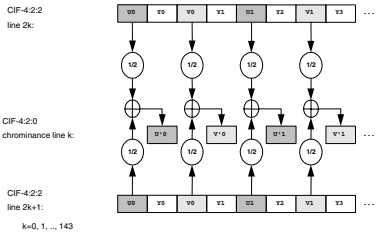


Fig. 9. CIF 4:2:2 to 4:2:0 conversion

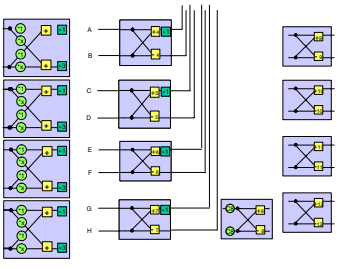


Fig. 10. CIF 4:2:2 to 4:2:0 conversion

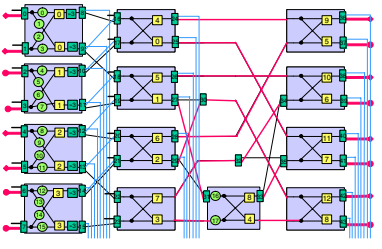


Fig. 11. Mapped Network

the luminance and chrominance data in a block-wise manner. The task of the IDCT is to re-transform the inverse quantized luminance and chrominance coefficients block-wise into the spatial domain. The input coefficients are provided as 12bit values by the control processor.

Obviously the IDCT depicted in Figure 8 is the reversed graph of the FDCT. Both consist of the same amount of operators, and both implementations use a similar operator network. Therefore, we map both graphs onto one common operator network. As a result, we get a re-configurable operator network. That means, we can use the operator network for encoding (FDCT) and decoding (IDCT) of a video image.

Lee [12] introduces an algorithm to split the N point FDCT/IDCT into two (N/2) point FDCTs/IDCTs. Therefore, to realize a 2D-FDCT/2D-IDCT we need eight instances of the FDCT/IDCT operator network. The throughput remains unchanged for the 2D-FDCT/2D-IDCT implementation.

Another algorithm is the video conversion CIF 4:2:2 to 4:2:0 (refer to Figure 9). The conversion CIF 4:2:2 to 4:2:0 is calculated by using the average value of the chrominance values. That means, previously, there are eight and after the conversion there are four values.

If we investigate the FDCT/IDCT network, part of the structure is similar to the CIF algorithm. We can use the second row to calculate the CIF conversion. Figure 10 shows the FDCT/IDCT network with the wires for the CIF algorithm. We place multiplexors before the operators, and routers at the output of the four

relevant lines after the operators. Thus, we fold the CIF algorithm into the FDCT/ IDCT network and receive the complete mapped network as depicted in Figure 11.

6 Results

Current implementations of the MACT architecture run on Xilinx's Virtex 400E FPGA. We have implemented all necessary basic components of the architecture to realize the described example.

Table 1. Logic utilization of the example for a Virtex 400E

Logic Utilization	used	total	perc.
No. of Slice FF	1897	9600	19%
Total no. 4-inp LUTS	1865	9600	19%
No. used as logic	1865	2393	78%
No. used as route-thru	528	2393	22%
Number of bonded IOBs	69	158	43%
IOB Flip Flops	21	69	30%
Number of GCLKs	1	4	25%
Number of GCLKIOBs	1	4	25%

Table 2. Logic distribution of the example for a Virtex 400E

Logic Distribution	used	total	perc.
No. of occupied Slices	1999	4800	41%
No. of Slices containing only related logic	1999	1999	100%
No. of Slices containing unrelated logic	0	2166	0%

We are analyzing the behavior of the system based on our own library. Additionally, we simulate the system. The high level synthesis produces appropriate code. The critical path of our example design needs 87 clock cycles. One cycle is 40 ns long. Therefore, the clock frequency is 25 MHz.

As MACT is a deeply pipelined design, the latency can be understood as setup time, which delays the system start only once. The latency of the example is 3480 ns (87 cycles multiplied with 40 ns). Further, we measure the throughput as average time between two output signals to determine the system speed. In the example, the throughput is equal to the clock frequency 40 ns. Taking into account that we run with 25 MHz we can achieve 25 MBit/s per wire.

The logic utilization and distribution are depicted in Tables 1 and 2. We can see that only 41 % of slices are used on the Virtex 400E.

7 Conclusion and Outlook

In this paper, we have presented concepts to realize routers for the MACT architecture. The concepts include variants of data encoding and router characteristics. We have extended the existing high level synthesis for the MACT architecture to be able to cope with routers and multiplexors. Routers within the MACT architecture enable reconfiguration. They facilitate to use one implementation of MACT for different application areas. Thereby, the bit-serial MACT architecture provides configurable functionality on the level of arithmetic operations,

high throughput rates, cost effective bit-serial operator design and short configuration cycles. The example of the IDCT/FDCT algorithm from Chen/Wang [5,6] demonstrates this effectiveness.

Further, we have propose an approach for reconfiguration, which decentralizes the control mechanism and thus results in short wire length. Thus, we will focus on future high-speed applications, where wire delay times affect the maximum processing clock.

References

1. A. Rettberg, T. Lehmann, M. Zanella, and C. Bobda, "Selbststeuernde rekonfigurierbare bit-serielle Pipelinearchitektur," Deutsches Patent- und Markenamt, Dec. 2004, patent-No. 10308510.
2. A. Rettberg, M. C. Zanella, C. Bobda, and T. Lehmann, "A Fully Self-Timed Bit-Serial Pipeline Architecture for Embedded Systems," in *Proceedings of DATE*, Munich, Germany, 3 - 7 Mar. 2003.
3. F. Dittmann, A. Rettberg, T. Lehmann, and M. C. Zanella, "Invariants for Distributed Local Control Elements of a New Synchronous Bit-Serial Architecture," in *Proceedings of the Delta*, Perth, Australia, 28 - 30 Jan. 2004.
4. A. Rettberg, F. Dittmann, M. C. Zanella, and T. Lehmann, "Towards a High-Level Synthesis of Reconfigurable Bit-Serial Architectures," in *Proceedings of SBCCI*, Sao Paulo, Brazil, 8 - 11 Sept. 2003.
5. Z. Wang, "Fast algorithms for the discrete W transform and for the discrete Fourier transform," *IEEE Trans. Acoust., Speech, Signal Proc.*, vol. 32, Aug. 1984.
6. W. Chen, C. Smith, and S. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Transaction Commun.*, vol. COM-25, 1977.
7. R. A. Bittner and P. M. Athanas, "Wormhole Run-time Reconfiguration," in *Proceedings of the 1997 ACM/SIGDA FPGA*, Monterey, CA, USA, Feb. 1997.
8. R. A. Bittner, P. M. Athanas, and M. D. Musgrove, "Colt: An Experiment in Wormhole Run-Time Reconfiguration," in *Photonics East, Conference on High-Speed Computing, Digital Signal Processing, and Filtering Using FPGAs*, Boston, MA, USA, Nov. 1996.
9. B. Blodget, P. James-Roxby, E. Keller, S. McMillian, and P. Sundararajan, "A Self-reconfiguring Platform," in *Proceedings of FPL*, Lisbon, Portugal, Sept. 2003.
10. W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. Comput.*, vol. 36, no. 5, pp. 547–553, May 1987.
11. A. Rettberg, M. C. Zanella, T. Lehmann, and C. Bobda, "A New Approach of a Self-Timed Bit-Serial Synchronous Pipeline Architecture," in *Proceedings of RSP Workshop*, San Diego, CA, USA, 9 - 11 June 2003.
12. B. G. Lee, "A New Algorithm to Compute the Discrete Cosine Transform," in *IEEE Trans. ASSP-32*, Dec. 1984.

An FPGA-Based Parallel Accelerator for Matrix Multiplications in the Newton-Raphson Method*

Xizhen Xu[1], Sotirios G. Ziavras[1], and Tae-Gyu Chang[2]

[1] Department of Electrical and Computer Engineering,
New Jersey Institute of Technology,
Newark, NJ 07102, USA
ziavras@adm.njit.edu
[2] School of Electrical and Electronics Engineering,
Chung-Ang University,
Seoul 156-756, South Korea

Abstract. Power flow analysis plays an important role in power grid configurations, operating management and contingency analysis. The Newton-Raphson (NR) iterative method is often enlisted for solving power flow analysis problems. However, it involves computation-expensive matrix multiplications (MMs). In this paper we propose an FPGA-based Hierarchical-SIMD (H-SIMD) machine with its codesign of the Hierarchical Instruction Set Architecture (HISA) to speed up MM within each NR iteration. FPGA stands for Field-Programmable Gate Array. HISA is comprised of medium-grain and coarse-grain instructions. The H-SIMD machine also facilitates better mapping of MM onto recent multimillion-gate FPGAs. At each level, any HISA instruction is classified to be of either the communication or computation type. The former are executed by a controller while the latter are issued to lower levels in the hierarchy. Additionally, by using a memory switching scheme and the high-level HISA set to partition applications, the host-FPGA communication overheads can be hidden. Our test results show sustained high performance.

1 Introduction

It is not uncommon in power flow analysis to make a good initial guess regarding the solution, e.g., a hot or flat start[16]. Thus, the Newton-Raphson iterative method is often used in power flow problems because a good initial guess leads to desirable convergence properties. If we profile the code of the NR algorithm, we will find out that the most expensive computations are MMs. Real time solutions to power flow problems are absolutely essential in power grid configurations, operating management and contingency analysis. In this paper, an FPGA-based parallel computing architecture is proposed to speed up the MM component in the NR iterations.

* This work was supported in part by the US Department of Energy under grant DE-FG02-03CH11171.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 458–468, 2005.
© IFIP International Federation for Information Processing 2005

Multimillion-gate FPGAs can form promising hardware accelerators for conventional hosts, e.g., a workstation or an embedded microprocessor[1][2][13] [14][15]. The workstation-FPGA combination is popular for data-intensive applications due to high FPGA resource efficiency and flexible workstation control. However, the substantial communication and interrupt overheads between the workstation and the FPGAs is also becoming a major performance bottleneck that may prevent further exploitation of the performance benefits gained from the parallel FPGA implementation [3][14].

Specifically, the contributions of our work are: i) We explore the FPGA-based design space to accelerate MM computations in NR iterations. To this extent, a hierarchical multi-FPGA system is proposed where each FPGA works in the SIMD (Single-Instruction Multiple-Data) parallel-processing mode. Under SIMD, all processors execute the same instruction simultaneously but on different data. Due to task partitioning with different granularities at various levels, we can eliminate communication requests of the processing elements (PEs) within the H-SIMD machine if a block-based matrix multiplication algorithm is employed. ii) We employ a memory switching scheme to overlap communications with computations as much as possible at each level. The conditions to fully overlap communications with computations are investigated as well. This technique overcomes the FPGA interrupt overheads and the rather low speed of the PCI bus that connects our FPGA-based target platform to the host [7]. Thus, our proposed methodology makes it possible to synthesize a scheme that brings together the computing power of the workstation and the FPGAs seamlessly for the NR algorithm.

Many research projects have studied MM for reconfigurable systems [4][5][6]. [4] proposed scalable and modular algorithms. Yet the authors point out that the proposed algorithms still incur high configuration overhead and large-sized configuration files. [5] introduced a parallel block-based algorithm for MM with substantial results. Though their design is based on a host-FPGA architecture and pipelined operation control is employed as well, the interrupt overhead from the FPGA to the host is not taken into consideration for a workstation host. Hence, [4][5] can not be used to accelerate MM in the NR method. [6] concluded that FPGAs can achieve higher performance with less memory and bandwidth than a commodity processor.

The rest of this paper is organized as follows. Section 2 analyzes the NR iterative method, and presents the H-SIMD machine design and its memory switching scheme. Section 3 presents the HISA instruction set for NR and analyzes workload balancing for MM across H-SIMDs different layers. Section 4 shows implementation results and a comparative study with other works. Section 5 draws conclusions.

2 Multi-layered H-SIMD Machine and Newton-Raphson Method

2.1 Newton-Raphson Iterative Method

The NR method employs Taylors series expression for a function with two or more variables [16]. It replaces the Gauss-Seidel method which is characterized

```
Initialization & flat start;
X1_matrix= alpha*transpose(A);
do{
   X0_matrix = X1_matrix;
   multiply_matrix(A_matrix, X0_matrix, temp1_matrix);
   multiply_minus(2*I, temp1_matrix, temp2_matrix);
   multiply_matrix(X0_matrix, temp2_matrix, X1_matrix);
}while( ||X1_matrix - X0_matrix|| > 0.000001);
```

Fig. 1. The pseudo-code for the NR iterations

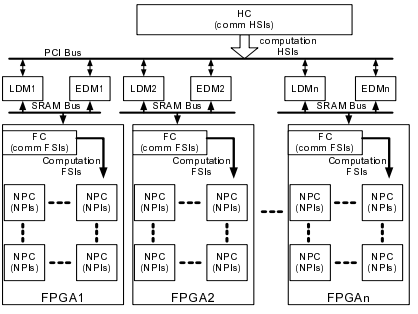


Fig. 2. H-SIMD machine architecture

by slower convergence. The nonlinear Newton-Raphson-type iteration for finding the reciprocal $1/A$ of matrix A is $X(k+1) = X(k)(2I - AX(k))$, where k is the iteration number and $X(0)$ is the initial guess for A^{-1}. The iterative technique proceeds until the sum of the absolute values of the off-diagonal elements in $AX(k)$ is less than ϵ, where ϵ is the required accuracy. The convergence rate is determined by the initial choice of $X(0)$. The process converges if and only if all eigenvalues of $I - AX(0)$ have absolute value less than one. Convergence, when it occurs, is generally quadratic. An improvement can be made so that the algorithms convergence is cubic. The pseudo-code for the NR algorithm is shown in Fig. 1. We can tell that two matrix multiplications are needed per iteration. Hence, our H-SIMD machine is designed for the acceleration of MM in the NR iterations.

2.2 H-SIMD Architecture

The H-SIMD control hierarchy is composed of three layers: the host controller (HC), the FPGA controllers (FCs) and the nano-processor controllers (NPCs), as shown in Fig. 2. The HC encounters the coarse-grain host SIMD instructions (HSIs) in the application program, which are classified into host-FPGA communication HSIs and time-consuming computation HSIs. The HC executes the communication HSIs only and issues computation HSIs to FCs. Inside each FPGA, the FC further decomposes the received computation HSIs into a sequence of medium-grain FPGA SIMD instructions (FSIs). The FC runs them in a manner similar to the HC: executing communication FSIs and issuing the computation FSIs to the nano-processor array. The NPCs finally decode the received computation FSIs into fine-grain nano-processor instructions (NPIs) and sequence their execution. Due to the difference between computation instructions and communication instructions at all levels, the H-SIMD machine configures one of the FPGAs as the master FPGA which sends an interrupt signal back to the HC once the previously executed computation HSI has been completed. Similarly, one NP within each FPGA is configured as the master NP

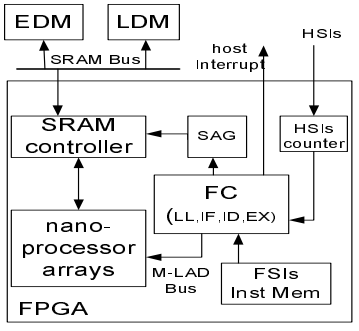


Fig. 3. HC-level memory switching scheme

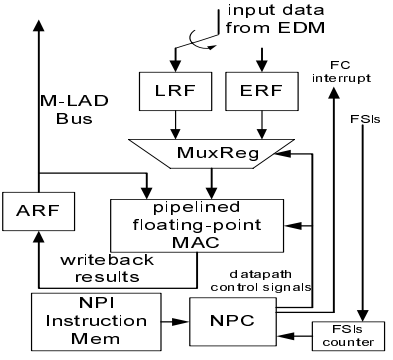


Fig. 4. Nano-processor datapath and control unit

that sends an interrupt signal back to its FC so that a new computation FSI can be executed.

The communication overhead between the host and the FPGAs is very high primarily due to the nature of the non-preemptive operating system on the workstation. Based on tests in our laboratory, the one-time interrupt latency for a Windows-XP installed workstation running the PCI bus at 133MHz is about 1.5 ms. This penalty is intolerable in high-performance computing [14]. Thus, a design objective of the H-SIMD machine is to reduce the interrupt overheads. A memory switching scheme has been applied successfully before in [5]. However, they did not specify the conditions to fully overlap communications with computations, a focus of our study. [14] studied such conditions but for another application problem.

The HC-level memory switching scheme is shown in Fig. 3. The SRAM banks on the FPGA board are organized into two functional memory units: the execution data memory (EDM) and the loaded data memory (LDM). Both are functionally interchangeable. At one time, the FCs access EDMs to fetch operands for the execution of received computation HSIs while LDMs are referenced by the host for the execution of communication HSIs. When the FCs finish their received computation HSI, they will switch between EDM and LDM to begin a new iteration. The FC is a finite-state machine responsible for the execution of the computation HSI. The FCs have access to the NP array over a modified LAD (M-LAD) bus. The LAD bus was originally developed by the Annapolis Micro Systems company for our target board and was used for on-chip memory references [7]. The M-LAD bus controller is changed from the PCI controller to the FCs. The HSI counter is used to calculate the number of finished computation HSIs. The SRAM address generator (SAG) is used to calculate the SRAM load/store addresses in EDM banks. The FC is pipelined and sequentially traverses the pipeline stages LL (Loading LRFs), IF (Instruction Fetch), ID (Instruction Decode) and EX (execution). The transition condition from EX to LL is triggered by the master NPs interrupt signal. The interrupt request/response

latency is one cycle only as opposed to the tens of thousands of cycles between the host and FPGAs, thus enhancing the H-SIMDs performance.

The nano-processor array forms the customized execution units in the H-SIMD machine datapath. Each nano-processor has three large-sized register files: the load register file (LRF), the execution register file (ERF) and the accumulation register file (ARF), as shown in Fig. 4. Both the LRFs and ERFs work in a "memory" switching scheme, similarly to the LDMs and EDMs. The ERFs are used for the execution of computation FSIs while the LRFs are referenced by the communication FSIs at the same time. The computation results are accumulated in the ARFs which can be accessed by the FCs.

3 HISA and Task Partitioning for MM

3.1 HISA: Instruction Set Architecture for MM

Similar to the approach for PC clusters in [10], we suggest here that an effective instruction set architecture (ISA) be developed at each layer for each application domain. The HC is programmed by host API (Application Programming Interface) functions for the FPGA board. They can initialize the board, configure the FPGAs, reference the on-board/on-chip memory resources and handle interrupts. We present here the tailoring of HSIs for a block-based MM algorithm. More specifically, we assume the problem C=A*B, where A, B, and C are NxN square matrices. When N becomes large, block matrix multiplication is used that divides the matrix into smaller blocks to exploit data reusability. Due to limited space here, refer to [9] for more details about block-based MM.

In the H-SIMD machine, only a single FPGA or NP is employed to multiply and accumulate the results of one block of the product matrix at the HC and FC levels, respectively. Coarse-grain workloads can keep the NPs busy on MM computations, while the HC and FCs load operands into the FPGAs and NPs sequentially. This simplifies the hierarchical design of the architecture and eliminates the need for inter-FPGA and inter-NP communications. Based on the H-SIMD architecture, the HC issues N_hxN_h sub-matrix blocks for all the FPGAs to multiply. N_h is the block matrix size for the HSIs. We have three HSIs here: i)$host_matrix_load(i, S_{LDM}, N_h)$; ii)$host_matrix_store(i, S_{LDM}, N_h)$; iii)$host_matrix_mul_accum(H_A, H_B, H_C, N_h)$. The first two HSIs are the communication instructions while the third one is the computation instruction.

The FC is a finite state machine in charge of executing the computation HSI. It decomposes *host_matrix_mul_accum* of size N_hxN_h into FSIs of size N_fxN_f, where N_f is the sub-block matrix size for the FSIs. Enlisted is the same block matrix multiplication algorithm as the one for the HC. The code for *host_matrix_mul_accum* is pre-programmed by FSIs and stored into the FC instruction memory. The FSIs are 32-bit instructions with mnemonics as follows: i)$FPGA_matrix_load(i, S_{LRF}, N_f)$; ii)$FPGA_matrix_store(i, S_{ARF}, N_f)$; iii)$FPGA_matrix_mul_accum(F_a, F_b, F_c, N_f)$. They are in charge of the communications and computations at the FPGA level.

The NPIs are designed for the execution of the computation FSI. The code for *FPGA_matrix_mul_accum* is pre-programmed with NPIs and stored into the NPC instruction memory. There is only one NPI to be implemented: the floating-point multiply accumulation $NP_MAC(R_a, R_b, R_c)$, where R_a, R_b, and R_c are registers for $R_c = R_a * R_b + R_c$. The NPI code for computation FSIs needs to be scheduled to avoid data hazards. They occur when operands are delayed in the addition pipeline whose latency is L_{adder}. Thus, the condition to avoid data hazards is $N_f^2 > L_{adder}$, which can be easily met.

3.2 Analysis of Task Partitioning

The bandwidth of the communication channels in the H-SIMD machine varies greatly. Basically, there are two interfaces in the H-SIMD machine: a PCI bus of bandwidth B_{pci} between the host and the FPGAs; and the SRAM bus of B_{sram} between the off-chip memory and the on-chip nano-processor array. The HSI parameter N_h is chosen in such a manner that the execution time $T_{host_compute}$ of *host_matrix_mul_accum* is greater than $T_{host_i/o}$ which is the sum of the execution time T_{HSI_COMM} of all the communication HSIs and the master FPGA interrupt overhead T_{fpga_int}. If so, the communication and interrupt overheads can be hidden. Let us assume that there are q FPGAs of p nano-processors each. Specifically, the following lower/upper bounds hold for matrix multiplication:

$$T_{host_compute} > \tau * N_h^3/p$$

$$T_{host_i/o} < T_{HSI_COMM} * q + T_{fpga_int} = 4 * b * N_h^2/B_{pci} * q + T_{fpga_int}$$

where τ is the nano-processor cycle time and b is the width in bits of each I/O reference. Simulation results in Fig. 5 show that the HSI computation and I/O communication times vary with N_h, p and q for b=64 and τ=7 ns. With increases in the block size of HSIs, the computation time grows in a cubic manner and yet the I/O communication time grows only quadratically, which is exploited by the H-SIMD machine. This means that the host may load the LDMs sequentially while all the FPGAs run in parallel the issued HSI.

For FC-level N_fxN_f block MM, we tweak N_f to overlap the execution time $T_{FPGA_compute}$ of *FPGA_matrix_mul_accum* with the sum $T_{FPGA_i/o}$ of the execution times $T_{NP_i/o}$ of all the communication FSIs and NP interrupt overheads T_{NP_int}. The following lower/upper bounds hold:

$$T_{FPGA_compute} > \tau * N_f^3$$

$$T_{FPGA_i/o} < T_{NP_i/o} * p + T_{NP_int} = 4 * b * N_f^2/(B_{sram} * N_{bank}) * p + T_{NP_int}$$

N_{bank} is the number of available SRAM banks for each FPGA. This condition can be easily met [14]. More SRAM banks can provide higher aggregate bandwidth to reduce the execution times of the communication FSIs. By using the above analysis of the execution time, we explored the design space for the lower bound on N_h and N_f, respectively. On the other hand, the capacity of the on-board and on-chip memories defines the upper bounds on N_h and N_f. For each FPGA on

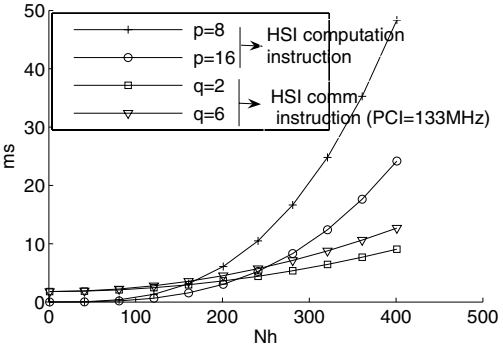


Fig. 5. Execution times of the computation and communication HSIs as a function of N_h, p and q

MM operations: $4*r*N_h^2*b < C_{sram}*N_{bank}$ and $4*r*N_f^2*b < C_{on-chip}$, where C_{sram} represents the capacity of one on-board SRAM bank; $C_{on-chip}$ represents the on-chip memory capacity of one FPGA; r stands for the redundancy of the memory systems, so r=2 for our memory switching scheme. In summary, N_h and N_f are upper-bound by $\sqrt{C_{SRAM}*N_{bank}/(8*b)}$ and $\sqrt{C_{on-chip}/(8*b)}$, respectively.

4 Implementation and Experimental Results

The H-SIMD machine was implemented on an Annapolis Wildstar II PCI board containing two Xilinx XC2V6000-5 Virtex-II FPGAs [7]. We used the Quixilica FPU [8] to build the NPs floating point MAC. In our design environment, ModelSim5.8 and ISE6.2 are enlisted as development tools. The Virtex-II FPGA can hold up to 16 NPs running at 148MHz. Broadcasts of FSIs to the nanoprocessor array are pipelined so that the critical path lies in the MAC datapath. The 1024x1024 MM operation was tested. The block size N_f of the FSIs was set to 8. The test results break down into computation HSIs, host interrupt overhead, PCI reference time, and initialization and NP interrupt overhead, as shown in Fig. 6. We can tell that the performance of the H-SIMD machine depends on the block size N_h. When N_h is set to 64, the frequent interrupt requests to the host contribute to performance penalty. When N_h is set to 128, the computation time does not increase long enough to overlap the sum of the host interrupt overhead and the PCI sequential reference overheads. If N_h is set to 512, there is long enough computation time to overlap the host interrupt. However, the memory switching scheme between the EDMs and LDMs does not work effectively because of the limited capacity of the SRAM banks, which results in penalties from both host interrupts and PCI references. If N_h is set to 256, the H-SIMD pipeline is balanced along the hierarchy such that the total execution time is very close to the peak performance $T_{peak} = N^3 * \tau/(p*q)$, where all the nanoprocessors work in parallel. We can sustain 9.1 GFLOPS, which is 95% of the

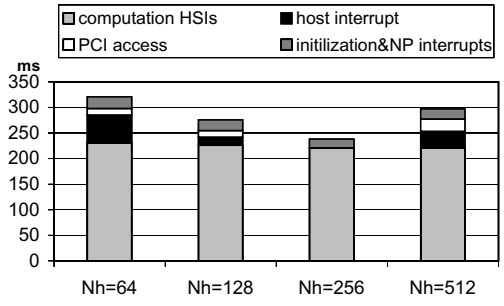


Fig. 6. 1024x1024 MM execution time as a function of N_h

peak performance. The execution overhead on the H-SIMD machine comes from the LDM and LRF initializations and the nano-processor interrupt to the FCs.

For arbitrary sizes of square matrices, a padding technique is employed to align the sizes to multiples of N_f because *FPGA_matrix_mul_accum* works on N_fxN_f matrices. N_f is set to 8 during the test. Let A and B be square matrices of size NxN. If N is not a multiple of eight, then both the A and B input matrices are padded up to the nearest multiples of eight by the ceiling function. Table 1 presents the test results for different cases. For matrices of size less than 512, the H-SIMD machine is not fully exploited and does not sustain high performance. For a large matrix ($N > 512$), the H-SIMD machine with two FPGAs can achieve about 8.9 GFLOPS on the average.

Table 2 compares the performance of our H-SIMD machine with that of previous work on FPGA-based floating-point matrix multiplications [4][5]. Their designs were implemented on Virtex II Pro125 containing 55,616 Xilinx slices as opposed to our Virtex II 6000 FPGA that contains 33,792 slices. We scaled up the H-SIMD size to match the resources in the Virtex II Pro125. After ISE place and route, 26 NPs can fit into one Virtex II Pro125 running at 180MHz and hence achieve the peak performance of 9.36GFLOPS per FPGA. The H-SIMD running frequency can be further increased if optimized MACs are enlisted. [4][5] presented a systolic algorithm to achieve 8.3 GFLOPS and 15.6 GFLOPS on a single XC2VP125 FPGA. However, the H-SIMD machine can be used as a computing accelerator for the workstation when the NR algorithm is implemented. In con-

Table 1. Execution times of MM for various test cases

Matrix size	H-SIMD machine(ms)	GFLOPS
200	7	2.28
400	18	7.111
600	50	8.683
1024	238	9.023
2048	1900	9.042
4000	14100	9.078

Table 2. Performance comparisons between H-SIMD and other works for a Virtex II Pro125 FPGA

	H-SIMD	[4]	[5]
Frequency	180	200	200
Number of PEs	26	24	39
GFLOPS	9.36	8.3	15.6
Hide interrupt overhead	Yes	No	No
Size of configuration files (MB/100 cases)	5	500	500

Table 3. Cost-performance comparison of the H-SIMD machine and the Xeon processor

System	**Transistors (millions)**	**Execution time T(ms)**	**VLSI Cost**	**Speedup**
2.8GHz	286	3.9	1	1
H-SIMD (2 FPGAs)	700	1.9	0.58	2.05

trast, the systolic approach does not fit into this computing paradigm because of the FPGA configuration overheads and the large size of configuration files.

The H-SIMD performance also compares favorably to that of state-of-the-art general purpose processors. The Intel Math Kernel Library (Intel MKL) contains the BLAS implementation that has been highly optimized for Intel processors. For double-precision general-purpose matrix multiplication (DGEMM), a 2.8 GHz Xeon with 512 KB L2 cache achieves 4.5 GFLOPS [11]. The time-consuming computations in the NR algorithm correspond to MMs. Thus, the H-SIMD machine can speed up the NR method by a factor of 1.9. A cost-performance analysis of the H-SIMD machine and a Xeon processor is in order now. The ten million system gates in the Virtex II FPGA consume about 400 million transistors. The H-SIMD machine built on the Annapolis board contains two Virtex II FPGAs. Thus, our current implementation of the H-SIMD machine employs roughly 700 million transistors. On the other hand, a 2.8GHz Xeon processor is comprised of about 286 million transistors [12]. For 2048x2048 MM on IEEE-754 double-precision numbers, it takes 3.9s on a Xeon processor as opposed to 1.9s on H-SIMD. According to a widely used VLSI complexity model, the cost C of implementing an algorithm is defined as $C = A * T^2$, where A is the chip area and T is the execution time. The chip area is directly proportional to the number of transistors, so we substitute in the cost equation the latter for the former. The VLSI cost and speedup results in Table 3 are normalized with respect to the Xeon processor. The H-SIMD machine can speedup the MM by a factor of two with only about half the VLSI cost.

5 Conclusions

In this paper, we analyzed the NR iteration algorithm for power flow problems and designed an FPGA-based MM accelerator for the host workstation. The proposed multi-layered H-SIMD machine paired with an appropriate multi-layered HISA software approach is effective for the host-FPGA architecture and can be synthetically used to speed up MM in NR iterations. To yield high performance, task partitioning is carried out at different granularity levels for the host, the FPGAs and the nano-processors. If the parameters of the H-SIMD machine are chosen properly, the memory switching scheme is able to fully overlap communications with computations. In our current implementation of matrix multiplication, a complete set of HISA for this application was developed. Its good performance was demonstrated. More recently introduced FPGAs, e.g, XC2VP125, could improve performance even further.

References

1. M. J. Wirthlin, B. L. Hutchings and K. L. Gilson: The Nano Processor: a Low Resource Reconfigurable Processor. Proc. IEEE FPGAs Custom Comput. (March 1994) 22-30
2. X. Wang and S. G. Ziavras: Performance Optimization of an FPGA-Based Configurable Multiprocessor for Matrix Operations. IEEE Intern. Conf. Field-Programmable Tech., Tokyo, (Dec. 2003)
3. P. H. W. Leong, C. W. Sham, W. C. Wong, H. Y. Wong, W. S. Yuen and M. P. Leong: A Bitstream Reconfigurable FPGA Implementation of the WSAT Algorithm. IEEE Trans. VLSI Syst., Vol. 9, No. 1, (Feb. 2001)
4. L. Zhuo and V. K. Prasanna: Scalable and Modular Algorithms for Floating-Point Matrix Multiplication on FPGAs. 18th Intern. Parallel Distr. Proc. Symp., (April 2004)
5. Y. Dou, S. Vassiliadis, G. K. Kuzmanov and G. N. Gaydadjiev: 64-bit Floating-Point FPGA Matrix Multiplication. 2005 ACM/SIGDA 13th Intern. Symp. FPGAs, (Feb. 2005)
6. K. D. Underwood and K. S. Hemmert: Closing the Gap: CPU and FPGA Trends in Sustainable Floating-Point BLAS Performance. IEEE Symp. Field-Progr. Custom Comput. Mach., (April 2004)
7. Wildstar II Hardware Reference Manual, Annapolis Microsystems, Inc, Annapolis, MD, (2004)
8. Quixilica Floating Point FPGA Cores Datasheet, QinetiQ Ltd, (2004)
9. R. Schreiber: Numerical Algorithms for Modern Parallel Computer Architectures. Springer-Verlag, New York, NY, (1988), 197–208
10. D. Jin and S. Ziavras: A Super-Programming Approach for Mining Association Rules in Parallel on PC Clusters. IEEE Trans. on Parallel Distr. Systems, Vol. 15, No. 9, (Sept. 2004)
11. Performance Benchmarks for Intel Math Kernel Library, Intel Corporation White Paper, (2003)
12. http://www.intel.com/pressroom/kits/quickreffam.htm#Xeon
13. X. Wang and S.G. Ziavras: Parallel LU Factorization of Sparse Matrices on FPGA-Based Configurable Computing Engines. Concurrency and Computation: Practice and Experience, Vol. 16, No. 4, (April 2004) pp.319–343

14. X. Xu and S.G. Ziavras: A Hierarchically-Controlled SIMD Machine for 2D DCT on FPGAs. IEEE International Systems-On-Chip Conference, Washington, D.C., (Sept. 2005)
15. X. Wang and S.G. Ziavras: Parallel Direct Solution of Linear Equations on FPGA-Based Machines. Workshop on Parallel and Distributed Real-Time Systems (in conjunction with the 17th Annual IEEE International Parallel and Distributed Processing Symposium), Nice, France, (April 2003)
16. W. F. Tinney and C. E. Hart: Power Flow Solution by Newton's Method. IEEE Trans. AS, PAS-86, No. 11, (1967) 1449–1460

A Run-Time Partitioning Algorithm for RTOS on Reconfigurable Hardware*

Marcelo Götz[1], Achim Rettberg[2], and Carlos Eduardo Pereira[3]

[1] Heinz Nixdorf Institute,
University of Paderborn, Germany
mgoetz@uni-paderborn.de
[2] C-LAB,
University of Paderborn, Germany
achim@c-lab.de
[3] Departamento de Engenharia Eletrica,
UFRGS - Universidade Federal do Rio Grande do Sul - Brazil
cpereira@eletro.ufrgs.br

Abstract. In today's system design, reconfigurable computing plays more and more an important role. By the extension of reconfigurable devices like FPGAs with one or more CPUs new challenges in system design should be solved. These new hybrid FPGAs (e.g. Virtex-II Pro™), provides a hardcore general-purpose processor (GPP) embedded into a field of programmable gate arrays. Furthermore, they offer partial reconfiguration. Therefore, those hybrid FPGAs are very attractive for implementation of run-time reconfigurable embedded systems. However, most of the efforts in this field were made in order to apply these capabilities at application level, leaving to the Operating System (OS) the provision of the necessary mechanisms to support these applications. In this paper, an approach for run-time reconfigurable Operating System, which takes advantage of the new hybrid FPGAs to reconfigure itself based on online estimation of application demands, is presented. Especially run-time assignment and reconfiguration of OS services over hybrid architecture are discussed. The proposed model uses a 0-1 Integer programming strategy for assigning OS components over hybrid architecture, as well as an alternative heuristic algorithm for it. Furthermore, the evaluation of the reconfiguration costs are presented and discussed.

1 Introduction

Nowadays, the usage of Field Programmable Gate Array (FPGA) in the field of Reconfigurable Computing (RC) has become widely used. In particular the capability of a FPGA to be run-time reprogrammed makes its use for reconfigurable systems very attractive Even more attractive is the emerging hybrid

* This work was developed in the course of the Special Research Initiative 614 - Self-optimizing Concepts and Structures in Mechanical Engineering - University of Paderborn, and was published on its behalf and funded by the Deutsche Forschungsgemeinschaft.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 469–478, 2005.
© IFIP International Federation for Information Processing 2005

FPGAs, which has a hardcore or softcore general purpose processor (GPP) surrounded by a large field of reprogrammable logic. These new components open several interesting possibilities to design reconfigurable architectures for Systems on Chip (SoC). [1].

One of the challenges of our research is to provide support for run-time reconfigurable architectures, which may be used for self-optimizing systems. In dynamic environments, where application requirements may dynamically change, the concept of reconfigurable operating systems appears, which is emerging as new research field.

Differently from the normal approach where the design of such operating system (OS) is done offline, the proposed approach suggests the use of new partial reconfigurable architectures in order to support the development of a hardware/software reconfigurable operating system [2]. In this proposed architecture, the Real-Time Operating System (RTOS) is capable to adapt itself to current application requirements, tailoring the RTOS components for this purpose. Therefore, the system continuously analyze the requirements and reconfigure the RTOS components at the hybrid architecture optimizing the use of system resources. Hence, the system is capable to decide on-the-fly which RTOS components are needed and also to which execution environment (CPU or FPGA) they will be assigned.

The paper focuses on an online partitioning algorithm for a real-time operating system services, which tries to minimize the whole resource utilization and to reduce the reconfiguration costs.

The remaining of the paper is organized as follows: Section 2 presents a brief state-of-the-art analysis regarding hardware implementation of OS services their flexibilities. Then, section 3 shortly presents the architecture used. Section 4 presents the system formulation using 0-1 Integer Programming (BIP) and the reconfiguration costs evaluation. An analysis of the run-time execution of this evaluations, with an heuristic algorithm for the components assignment problem are presented in section 5. Section 6 presents some evaluation results using MATLAB to compare the proposed heuristic algorithm with the original one presented in [3]. Finally, some conclusions and future work are shown in section 7.

2 Related Work

The idea of implementing OS services in hardware is not new. Several works in the literature, [4],[5],[6],[7] and [8]. show that hardware implementation may significantly improve performance and determinism of RTOS functionalities. The overhead imposed by the operating system, which is carefully considered in embedded systems design due to usual lack of resources, is considerably decreased by having RTOS services implemented in hardware. However, up to now all approaches have been based on implementations that are static in nature, that means, they do not change during run-time even when application requirements may significantly change.

In the field of reconfigurable computing, reconfiguration aspects have been concentrated at application level (see [9], [10] and [11]). At the OS level the research are limited to provide run-time support for those applications (see [6], [12] and [13]).

In the approach presented in this paper we expand the usage of those concepts and the hardware support to the OS level. Additionally, based on the state-of-the-art analysis, for a self-optimized reconfigurable RTOS none such similar approach has been proposed yet.

3 Basic Architecture

Our target architecture is composed of one CPU, configurable logic elements (FPGA), memory and a bus connecting the components. Most of these elements are provided in a single CHIP, such as the Virtex II Pro™, from Xilinx company. The RTOS services that are able to be reconfigured are stored on an external SDRAM chip in two different implementations: as software object and as FPGA configuration bitstream.

Abstractly, the system can be seen as presented in Figure 1. The target RTOS provide services to the application as a set of hardware and software components. These components may, during run-time, be reallocated (reconfigured) over the hybrid architecture in order to better use the available resources and still meet the application requirements.

Our system concept has a similar approach than a microkernel RTOS, as it is being the concept adopted by most RTOSs. Thus, just absolutely essential core operating system functions are provided by the kernel (which are fixed and can not be reconfigured). The other functionalities (services) are provided by components attached at the kernel. However, these components are reallocated during run-time in order to meet the application requirements and the resource usage constraints.

The usage of microkernel also incorporates the nature advantage of flexibility and extensibility (among others) of it, which is very desired in our case in order to better perform the reconfigurability aspects. Nevertheless, it has the disadvantage to slow down the performance due to the increased necessity of messages changes by the components. Therefore, the Communication Layer presented in Figure 1 performs a efficient communication infrastructure in order to reduce this effect and to offer a transparent set of services to the application (independent of the allocation of the components). The details of the architecture is in its development phase will not be treated in the scope of this paper.

4 Problem Definition

The problem of assigning RTOS components over the two execution environments can be seen as a typical assignment problem. Therefore, we decided to model the problem using Binary Integer Programming (BIP) [14]. A set of available services is represented as $S = \{s_{i,j}\}$, where every service i has its implemen-

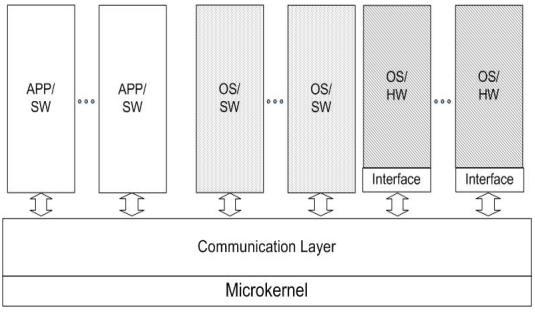


Fig. 1. Proposed microkernel based architecture

tation for CPU ($j = 1$) or FPGA ($j = 2$). Every component has an estimated cost $c_{i,j}$, which represents the percentage of resource from the execution environment used by this component. On the FPGA it represents the circuit area needed by the component and at CPU it represents the processor used by it. Note that these costs are not static, since the application demands are considered to be dynamic. This topic will be addressed later on in subsection 4.2.

4.1 OS Service Assignment

The assignment of a component to either CPU or FPGA is represented by the variable $x_{i,j}$. We say that $x_{i,j} = 1$ if the component i is assigned to the execution environment j, and $x_{i,j} = 0$ otherwise. As some of the components may not necessary be needed by the current application, they should neither be assigned to the CPU ($j = 1$) nor to the FPGA ($j = 2$). Therefore, to proper represent this situation we consider that this component should stay at memory pool ($j = 3$). As we are focusing on the resource utilization optimization between CPU and FPGA we define that a component i placed on the memory pool ($j = 3$) does not consume any resource ($c_{i,3} = 0$). The definition of a third assignment place for an OS component is more useful for reconfiguration costs estimation, that will be seen in section 4.2.

The resources are limited, which derive two constraints for our BIP formulation: the maximum FPGA area available (A_{max}) and the maximum CPU workload (U_{max}) reserved for the operating system. Thus, the total FPGA area (A) and total CPU workload (U) used by the hardware components and the software components, respectively can not exceed their maximums. These constraints are represented by

$$U = \sum_{i=1}^{n} x_{i,1} c_{i,1} \leq U_{max}, \qquad A = \sum_{i=1}^{n} x_{i,2} c_{i,2} \leq A_{max}$$

We also consider that a component i can be assigned just to one of the execution environment: $\sum_{j=1}^{3} x_{i,j} = 1$ for every $i = 1, ..., n$.

To avoid that one of the execution environment would have its usage near to the maximum, we specify a constraint to keep a balanced resource utilization (B) between the two execution environments: $B = |w_1 U - w_2 A| \leq \delta$. Where δ is the maximum allowed unbalanced resource utilization between CPU and FPGA. The weights w_1 and w_2 are used to proper compare the resource utilization between two different execution environments. If the resource used from an execution environment are not near to its maximum, it will have the capability to absorb some variation of the application demands. This characteristic are useful for real-time system in order to avoid the application to miss its deadlines due to workload transients. Note that this approach cannot guarantee hard real time constraints. However, for soft real-time systems it can be considered valid.

The objective function used to minimize the whole resource utilization is defined as

$$min\{\sum_{j=1}^{3}\sum_{i=1}^{n} c_{i,j}x_{i,j}\}$$

The solution of this BIP are the assignment variables $x_{i,j}$, which we define as being a specific system configuration: $\Gamma = \{x_{i,j}\}$.

4.2 Reconfiguration Costs

As is has been said in the section 4, the application requirements are considered to change over system life time. These modifications are represented by changes of the component costs $c_{i,j}$. This leads to the fact that a certain system configuration Γ may no longer be valid after application changes. Therefore, a continuously evaluation of the components partitioning is necessary. Whenever the systems reaches a unbalanced situation ($|w_1 U - w_2 A| > \delta$), the RTOS components should be reallocated in order to bring the system again in the desired configuration. In this situation, not just the new assignment problem need to be solved (Γ') again, but also the costs (time) necessary to reconfigure the system from Γ to Γ' need to be evaluated. This evaluation is necessary since we are dealing with real-time systems. Thus, we have a limited time available for reconfiguration activities.

The reconfiguration cost of every component represents the time necessary to migrate a component from one execution environment to the other one. Therefore, we need to specify for every possible migration of a component its correspondent cost. As it was shown in section 4.1, our model assumes three different environments ($j = \{1, 2, 3\}$). The definition of the environment $j = 3$ (memory pool) is necessary to proper represent the case where a new OS service arrives in the system. This happens when the application requires a service that is neither at CPU nor at FPGA available, but it is stored at the memory pool. The same is valid for a service that leaves the system (it is not more needed by the application). So, we define for a component i a 3×3 size migration costs matrix R_i. Let $R_i = \{r^i_{j,j'}\}$, where j and j' are the current and new execution environment of component i.

If $x_i = \{x_{i,1}, x_{i,2}, x_{i,3}\}$ and $x'_i = \{x'_{i,1}, x'_{i,2}, x'_{i,3}\}$ are the current and new assignment of the component i, then the complete reconfiguration cost K (total reconfiguration time) of the system is defined as:

$$K = \sum_{i=1}^{n} x_i^T R_i x'_i$$

In our current approach the migration costs associated which a component includes all necessary steps to remove a component from one execution environment to the other one. These steps represents the time to program the FPGA with a component or link the software component with in the CPU programm, translate the context between different execution environments (when necessary), and also read the component instance from memory pool.

5 Run-Time Analysis

As our operating system is being designed to support real-time applications, a deterministic behavior for service assignment and system reconfiguration need to be used in order to handle application time constraints.

5.1 Heuristic Algorithm for Assignment Problem

The solution of an BIP finds an optimal solution for the assignment problem. For a small set of components this approach is very suitable. However, it is too computationally complex to solve all problem sizes. Therefore, we are currently using an heuristic greedy based algorithm for this problem. The algorithm is composed by two parts. The first one creates two clusters (FPGA and CPU component sets) from the component set currently needed by the application. The second part improves the first solution towards the balance value (B) and the number of the components to be reconfigured (trying to reduce it). The next paragraphs will concentrate in second part of the algorithm and the details about the first phase can be seen in [3].

The solution given by the first part of the algorithm do not take into consideration the balancing constraint δ. So, there is no guarantee that it will provide a solution which fulfills this constraint. Moreover, it does not take into consideration the reconfiguration costs reduction. Therefore, a second algorithm is proposed that improves the balancing B in order to meet the δ constraint. It is based on Kernighan-Lin algorithms [15] and it aims to obtain a better balancing B than the first one by swapping pairs of components between CPU and FPGA. It also tries, to minimize the number of components being reconfigured in order to reduce the total reconfiguration cost. The algorithm receives as input the first assignment solution X which has $nc_1 = \sum_{i=1}^{n} x_{i,1}$ components assigned to CPU and $nc_2 = \sum_{i=1}^{n} x_{i,2}$ components assigned to FPGA. The maximum number of pairs that are possible to be swapped is defined as: $max_pairs = min(nc_1, nc_2)$.

By moving a component i, previously assigned to the CPU, to the FPGA ($\{x_{i,1} = 1; x_{i,2} = 0\} \Rightarrow \{x_{i,1} = 0; x_{i,2} = 1\}$), we have a new balancing B:

$B_{new} = |B_{current} - s_i|$, where $s_i = \{c_{i,1} + c_{i,2}\}$. Similarly, by moving a component i from FPGA to CPU, the new balancing B will be: $B_{new} = |B_{current} + s_i|$. Thus, swapping a pair of components o, p ($\{x_{o,1} = 1; x_{o,2} = 0\}; \{x_{p,1} = 0; x_{p,2} = 1\}$), the new balancing B is defined as: $B_{new} = |B_{current} - s_o + s_p|$. Similarly, $B_{new} = |B_{current} + s_o - s_p|$ if $\{x_{o,1} = 0; x_{o,2} = 1\}; \{x_{p,1} = 1; x_{p,2} = 0\}$. Additionally, we define G_{op} as the gain obtained in the balancing B by swapping a pair o and p of components: $G_{op} = B_{new} - B_{current}$. A gain below 0 means an improvement obtained in the balancing B.

The reconfiguration costs reduction is executed indirectly by means of reducing the number of components to be reconfigured. Therefore, a function $\Delta X = diff(X^a, X^b)$ (where $\Delta X = \{\delta x_i\}$, $X^a = \{x^a_{i,j}\}$ and $X^b = \{x^b_{i,j}\}$) is used to give the information if a component x_i has different allocation in X^a and X^b. Thus, the function $diff$ is defined as follows:

$$\delta x_i = \begin{cases} 1 & : \texttt{if } \{x^a_{i,1}; x^a_{i,2}\} \neq \{x^b_{i,1}; x^b_{i,2}\} \\ 0 & : \texttt{otherwise} \end{cases}$$

The algorithm starts trying to swap all possible pairs and storing the gain obtained by every try. It than chooses the one which provides the smallest gain. If this gain is bigger than or equal to zero, none swap is able to provide an improvement in the balancing B and the algorithm stops. Otherwise, the pair is

Algorithm 1. Balancing B improvement and reconfiguration cost reduction

$X^{init}_1 = \{x_{i,1}\}$ Initial assignment of CPU components
$X^{init}_2 = \{x_{i,2}\}$ Initial assignment of FPGA components
$X^{init} = X^{init}_1 \cup X^{init}_2$; $X^{new} = X^{init}$;
$B^{init} = |U^{init} - A^{init}|$; $B^{new} = B^{init}$;
$X^{orig} =$ *Current System Configuration* Γ
$m = max_pairs$ maximum number of pairs
for $k = 1$ to m **do**
  Find the pair o, p ($\{x_{o,1} = 1; x_{o,2} = 0\}$;$\{x_{p,1} = 0; x_{p,2} = 1\}$ or $\{x_{o,0} = 0; x_{o,2} = 1\}$;$\{x_{p,1} = 1; x_{p,2} = 0\}$) so that o and p are unlocked and G_{op} is minimal
  if $G_{op} < 0$ **then**
    Swap o and p and test it $\Rightarrow X^{try} = (X^{new}$ with o and p swapped)
    $\Delta X = diff(X^{orig}, X^{try})$
    if $\delta x_o = 0$ OR $\delta x_p = 0$ **then**
      Update the new configuration $\Rightarrow X^{new} = X^{try}$
      $B^{new} = B^{new} + G_{op}$
      if $\Delta X_o = 0$ **then** Lock o **end if**
      if $\Delta X_p = 0$ **then** Lock p **end if**
    else
      if $x^T_o R_o x'_o < x^T_p R_p x'_p$ **then** Lock o **else** Lock p **end if**
    end if
  end if
  if $G_{op} \geq 0$ OR all pairs are locked **then break**
end for
return X^{new}

swapped and locked according to some rules. If, at least, one of the components from the pair keeps its position in relation to the current system configuration, the pair swap is allowed. In addition, the component that preservers its position (or both) are locked (no longer a candidate to be swaped). However, if both components of the pair change their positions in relation to the current system configuration, no swap occurs. Moreover, just one component (which provides the smaller reconfiguration cost) is locked. This lock is necessary, otherwise the algorithm would not terminate. This process is then repeated until all pairs have been locked or no improvement can be obtained by any interchange. By applying those rules, the algorithms tries to reduce the numbers of components needed to be reconfigured. Also note that the algorithm does not terminate if the δ constraint is fulfilled. This enforce the search for more components (pairs) that could be kept in its current allocation solution.

The algorithm terminates by returning the new assignment solution X that provides a better (or at least an equal) balancing B than the solution provided by the first one. In addition, the number of components being reconfigured is reduced. The complexity of the balancing improvement algorithm is (worst case) $O(m^3)$, where m is the maximum number of pairs. This is due to the fact that we have one *for* loop (1 to m) where in each loop interaction the combination of all components assigned to the CPU with all components assigned to the FPGA are tested (m^2). The algorithm for balancing improvement is shown in Algorithm 1.

6 Experimental Results

For system evaluation of the run-time assignment problem, we made some simulations using MATLAB tool. The results achieved by the original balance improvement algorithm (published in [3]) and the improved one (presented in this paper) were compared. We generated a number of 100 different systems having randomly costs: $1\% \leq c_{i,1} \leq 15\%$ and $5\% \leq c_{i,2} \leq 25\%$; and fixed size: $n = 20$ components. The maximum FPGA resource was defined to be 100% ($A_{max} = 100$), as well as for the CPU ($U_{max} = 100$). The components assignment were calculated for every system using the 0-1 Integer Programming (optimal solution) and the heuristic algorithm (first and second one). The absolute difference cost ($|w_1 U - w_2 A|$) and the number of components being reconfigured achieved each version of the balancing improvement algorithm were compared for different values of δ (the resource usage balancing restriction): $(0.5, 1, 2, 3, 4, 5, 10, 20, 30, 40, 50\, and\, 60)$. The current system configuration considered was the previous random system generated.

The Figure 2 shows the results of the balance improvement achieved by the original algorithm (Heuristic-2a) and the optimized one (Heuristic-2b). The improvement made in both case were satisfactory. Note that the results achieved by Heuristic-2b, concerning the balance, are quite under the constraint δ. This is due to the fact that the Heuristic-2a algorithm still search for more pairs to be swapped, even with the δ constraint being fulfilled, in order to reduce the number of reconfigurations. This effect can be seen in Figure 3.

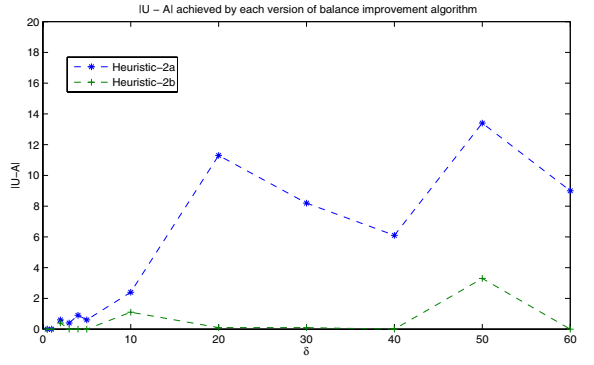


Fig. 2. Unbalance average for different δ constraints

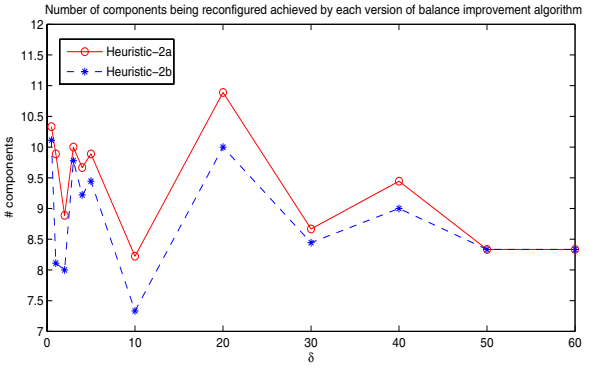


Fig. 3. Number of components being reconfigured for different δ constraints

7 Conclusions and Future Work

In this paper we have presented our investigation towards a run-time reconfigurable RTOS running over a hybrid platform, focusing in the OS service assignment and system reconfiguration. Looking at the related work, we are quite convinced that this is a novel approach for a self-optimized RTOS.

A shortly presentation of the concept of our architecture was also presented. The 0-1 Integer Programming model of the system and the reconfiguration cost evaluation have been presented. Additionally, considerations of a run-time execution of this technics, in order to support real-time applications have been discussed.

As a future work, the investigation of a proper OS components assignment algorithm which takes into consideration the application time constraints and the integration of the communication costs among the components are going to be made. Moreover, the schedule of the components reconfiguration using technics

of RTOS Scheduling, necessary to guarantee the application time requirements are going to be integrated.

References

1. Andrews, D., Niehaus, D., Ashenden, P.: Programming models for hybrid cpu/fpga chips. Computer - Inovative Thecnology for Computer Professionals (2004) 118–120 IEEE Computer Society.
2. Götz, M.: Dynamic hardware-software codesign of a reconfigurable real-time operating system. In: Proc. of ReConFig04, Mexican Society of Computer Science, SMCC (2004)
3. Götz, M., Rettberg, A., Pereira, C.E.: Towards run-time partitioning of a real time operating system for reconfigurable systems on chip. In: Proceedings of International Embedded Systems Symposium 2005, Manaus, Brazil (2005)
4. Lee, J., III, V.J.M., Ingstrm, K., Daleby, A., Klevin, T., Lindh, L.: A comparison of the rtu hardware rtos with a hardware/software rtos. In: ASP-DAC2003, (Asia and South Pacific Design Automation Conference). (2003) 6 Japan.
5. Kuacharoen, P., Shalan, M., Mooney, V.: A configurable hardware scheduler for real-time systems. In: Proc. of ERSA. (2003)
6. Kohout, P., Ganesh, B., Jacob, B.: Hardware support for real-time operating systems. In: International Symposium on Systems Synthesis, Proceedings of the 1st IEEE/ACM/IFIP International conference on HW/SW codesign and system synthesis (2003)
7. Lee, J., Ryu, K., III, V.J.M.: A framework for automatic generation of configuration files for a custom hardware/software rtos. In: Proc. of ERSA. (2002)
8. Lindh, L., Stanischewski, F.: Fastchart - a fast time deterministic cpu and hardware based real-time-kernel. In: EUROMICRO'91. (1991) 12–19 Paris, France.
9. Harkin, J., McGinnity, T.M., Maguire, L.P.: Modeling and optimizing run-time reconfiguration using evolutionary computation. Trans. on Embe.Comp.Sys. (2004)
10. Quinn, H., King, L.A.S., Leeser, M., Meleis, W.: Runtime assignment of reconfigurable hardware components for image processing pipelines. In: FCCM. (2003)
11. Mignolet, J.Y., Nollet, V., Coene, P., Verkest, D., Vernalde, S., Lauwereins, R.: Infrastructure for design and management of relocatable tasks in a heterogeneous reconfigurable system-on-chip. In: DATE. (2003)
12. Wigley, G., Kearney, D.: The development of an operating system for reconfigurable computing. In: FCCM. (2001)
13. Walder, H., Platzner, M.: A runtime environment for reconfigurable hardware operating systems. In: FPL. (2004)
14. Wolsey, L.A.: Integer Programming. Wiley-Interscience (1998)
15. Eles, P., Kuchcinski, K., Peng, Z.: 4. In: System Synthesis with VHDL: A Transformational Approach. Kluwer Academic Publishers (1998)

UML-Based Design Flow and Partitioning Methodology for Dynamically Reconfigurable Computing Systems

Chih-Hao Tseng and Pao-Ann Hsiung

Department of Computer Science and Information Engineering,
National Chung-Cheng University, Chiayi, Taiwan, R.O.C.
hpa@computer.org

Abstract. Dynamically reconfigurable computing systems (DRCS) provides an intermediate tradeoff between flexibility and performance of computing systems design. Unfortunately, designing DRCS has a highly complex and formidable task. The lack of tools and design flows discourage designers from adopting the reconfigurable computing technology. A UML-based design flow for DRCS is proposed in this article. The proposed design flow is targeted at the execution speedup of functional algorithms in DRCS and at the reduction of the complexity and time-consuming efforts in designing DRCS. In particular, the most notable feature of the proposed design flow is a HW-SW partitioning methodology based on the UML 2.0 sequence diagram, called *Dynamic Bitstream Partitioning on Sequence Diagram (DBPSD)*. To prove the feasibility of the proposed design flow and DBPSD partitioning methodology, an implementation example of DES (Data Encryption Standard) encryption/decryption system is presented in this article.

Keywords: UML, sequence diagram, partitioning, design flow, reconfigurable computing, FPGA, codesign.

1 Introduction

The acceleration of computing-intensive applications requires more powerful computing architectures. Continuing improvement in microprocessor has increased computing speed, however it is still slower than the speed required by the computing-intensive applications. Microprocessors provide high flexibility in executing a wide range of applications, but they suffer from limited performance. Application specific integrated circuits (ASICs) provide superior performance, but are restricted by limited set of applications. Thus, a new computing paradigm is called for. DRCS [1], [2] is a promising solution, which provides an intermediate tradeoff between flexibility and performance.

The work in this article is concerned with the development of a design flow and of related supporting tools for DRCS. The proposed design flow takes a UML-based application model and facilitates the co-synthesis and rapid prototyping of dynamically reconfigurable computing systems. The outputs of our

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 479–488, 2005.
© IFIP International Federation for Information Processing 2005

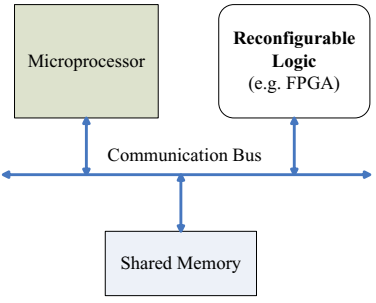


Fig. 1. Model of Reconfigurable Computing Architecture

design flow are the ready-to-run software application and hardware bitstreams for target platform, which is a dedicated FPGA (Field Programmable Gate Arrays) board connected to the host computer over the PCI (Peripheral Component Interconnect) interface. Furthermore, the primary focus of this article is on the hardware-software partitioning of UML models, which makes it different from previous researches.

Reconfigurable computing systems refer to systems, which contain a part of hardware that can change its circuits at run-time to facilitate greater flexibility without compromising performance [1]. The architecture of reconfigurable computing systems typically combine a microprocessor with reconfigurable logic resources, such as FPGA. An abstract model of such an architecture appears in Fig. 1. The microprocessor executes the main control flow of application and operations that cannot be done efficiently in the reconfigurable logic. The reconfigurable logic performs the computing-intensive parts of the application. The shared memory is used for communication between the microprocessor and the reconfigurable logic.

Unfortunately, designing these kinds of systems is a formidable task with a high complexity. Although many researches are ongoing in the academia [6]-[10] and industry, but the lack of mature tools and design flows discourage designers from adopting the reconfigurable computing technology. This leads to the need for a design flow especially easy for use by software engineers.

Issues encountered in constructing a design flow for dynamically reconfigurable computing systems are as follows.

- How to provide a HW-SW partitioning methodology, which is intuitional and easy for a software engineer?
- How to help the software engineer to synthesize the hardware bitstreams without much knowledge in digital hardware design?
- The communication between software and hardware is crucial. Thus, how must communication be designed easily, correctly, and efficiently?
- What kind of target hardware platform is appropriate for this design flow?

To solve the issues mentioned above, we develop a UML-based design flow and related supporting tools for the rapid application prototyping of dynamically

reconfigurable computing systems. The features of our proposed solutions are as follows:

- Supporting software-oriented strategy for co-synthesis as we start from a UML specification and identify parts which are to be implemented in reconfigurable hardware.
- Automatic synthesis of bitstreams for reconfigurable hardware.
- Automatic generation of software/hardware interfaces.
- Using commercial off-the-shelf FPGA within PCI card to facilitate the construction of target platform and using API (Application Programming Interface) for reconfiguration.

The article is organized as follows. In Sect. 2, we give a detailed discussion of the proposed design flow. Section 3 is the core of this article, where we present our partitioning methodology for UML models. In Sect. 4, some examples are presented to show the feasibility of the proposed design flow. In the last section, we conclude this article and introduce some future work.

2 The Design Flow

The proposed design flow, as shown in Fig. 2, is separated into three phases: *design and implementation of the system software model*, *hardware synthesis*, and *software synthesis*. C++ code and UML 2.0 diagrams such as the class diagram, sequence diagram, and state machine diagram are the inputs of the proposed design flow. Reconfigurable C++ application and bitstreams are outputs of our design flow. In Fig. 2, the elliptical boxes such as Class Diagram, XMI Documents, and Bitstreams represent the workproduct in each phase. The rectangular boxes such as Rhapsody 5.0, C++ Compiler, Forge, and XFlow represent commercial-off-the-shelf tools. The three-dimensional rectangular boxes such as SW/HW Extractor, SW Interface Synthesizer, and HW Interface Synthesizer represent tools developed by ourselves. Certainly, the target platform for verifying the proposed design flow is also constructed. Each phase of the proposed design flow and the target platform will be examined further in the following subsections.

2.1 Design and Implementation of System Software Model

In the *Design and Implementation of System Software Model* phase of the proposed design flow, we use the Rhapsody 5.0 tool to build the UML models, implement the detailed behaviors in C++, verify the functionalities of the system software model, and partition performance critical methods into hardware. After the model is constructed, XMI documents are generated by the XMI toolkit of Rhapsody 5.0. XMI documents use the XML format to store UML model information, and C++ code is also included. We then use our SW/HW Extractor to parse partitioning information from the XMI documents. The SW/HW Extractor searches the XMI documents to locate UML stereotype <<HW>>

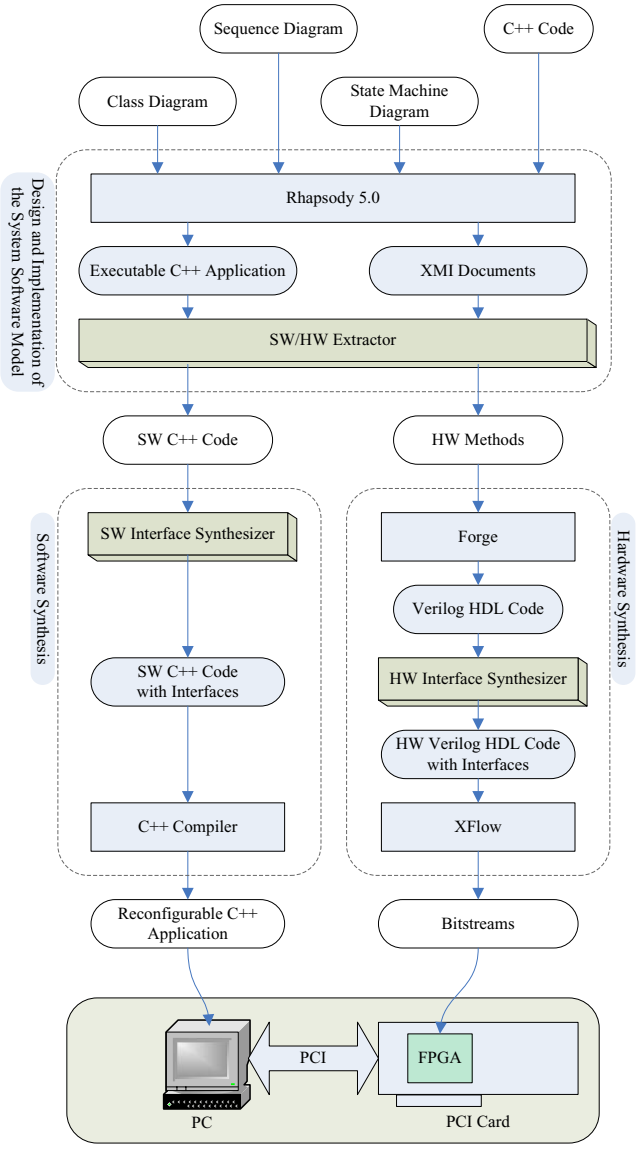


Fig. 2. Design Flow for Dynamically Reconfigurable Computing Systems

marked by user, then it extracts this portion of C++ method as HW method for post synthesis in the *Hardware Synthesis* phase. The SW C++ code is another output of SW/HW Extractor, which represents all of the C++ application code. This SW C++ code will be the input of the *Software Synthesis* phase. After the *Design and Implementation of System Software Model* phase, the design flow is split up into the *Hardware Synthesis* phase and the *Software Synthesis* phase.

The Unified Modeling Language (UML) [3] is a standard modeling language used in the software industry. In this work, we have chosen three diagrams from UML 2.0 for building the system model, namely class diagram, state machine diagram, and sequence diagram. Class diagrams are used to model the architecture of software application.

State machine diagrams describe the dynamic behavior of a class in response to external stimuli, such as event or trigger. There exists a gap between the state machine diagram and its implementation. According to [4], the state machine diagrams include many concepts such as states and transitions that are not present in most object-oriented programming languages, like Java or C++. In order to overcome this gap, we adopt Rhapsody [5] as our UML modeling tool. After drawing UML models in Rhapsody, the tool can generate Ada, C, C++, or Java code. Rhapsody provides an Object Execution Framework (OXF) [5], which enables state machine diagrams to be implemented in object-oriented programming languages.

Sequence diagrams show the interactions between classes in a scenario that is a partial system behavior of overall system specifications. In a sequence diagram, classes and objects are listed horizontally as columns, with vertical lifelines indicating the lifetime of the object over time. Messages are rendered as horizontal arrows between classes or objects and represent the communication between classes or objects.

2.2 Hardware Synthesis

The purpose of the *Hardware Synthesis* phase is to synthesize the hardware bitstreams which will be used by software applications, to perform some required computing-intensive operation. The HW Methods derived from SW/HW Extractor are the inputs of this phase. First, the Forge tool is used to transform the HW Methods into synthesizable Verilog HDL (Hardware Description Language) code. Secondly, the HW Interface Synthesizer adds PCI wrapper for the Verilog HDL code, then produces the HW Verilog HDL code with interface which enable communication from software. Finally, the XFlow tool will synthesize the HW Verilog HDL code with interface into the hardware bitstreams for execution in FPGA.

2.3 Software Synthesis

The purpose of the *Software Synthesis* phase is to build an executable C++ application, which is capable of invoking hardware methods on demand. The Software C++ code which was derived from the SW/HW Extractor is the input of this phase. Starting from this code, the Sofware Interface Synthesizer is used to synthesize code for accessing hardware methods. After that, the produced Software C++ code with interfaces is the final source code. Finally, this code is compiled by a C++ compiler to generate a ready-to-run C++ program, called Reconfigurable C++ Application. During the execution on the host processor, this Reconfigurable C++ Application can reconfigure required hardware method into FPGA for acceleration of the software execution.

2.4 Target Platform

Target platform allows system designers to verify the overall system behavior and to evaluate the overall system performance. Our target platform is a dedicated Xilinx Virtex-II FPGA (XC2V3000, 28,672 LUTs, at 40MHz) board connected to the host computer (Pentium 4 2.8GHz, 1GB RAM, Windows XP) over the PCI interface.

3 The DBPSD Partitioning Methodology

We propose a *Dynamic Bitstream Partitioning on Sequence Diagrams (DBPSD)*, which is a partitioning methodology based on the UML 2.0 sequence diagrams and includes *partitioning guidelines* to help designers make prudent partitioning decisions at the granularity of class methods. Sequence diagrams in UML 2.0 have been significantly enhanced for specifying complex control flows in one sequence diagram. As shown in the middle of Fig. 3, the most obvious changes are the three rectangular boxes, called *interaction fragments*. The five-sided box with labels such as **alt**, **opt**, or **loop** at the upper left-hand corner is the *interaction operator* of the interaction fragment.

The interaction operator gives the interaction fragment specific meaning. The **alt** operator denotes a conditional choice according to the evaluation results of

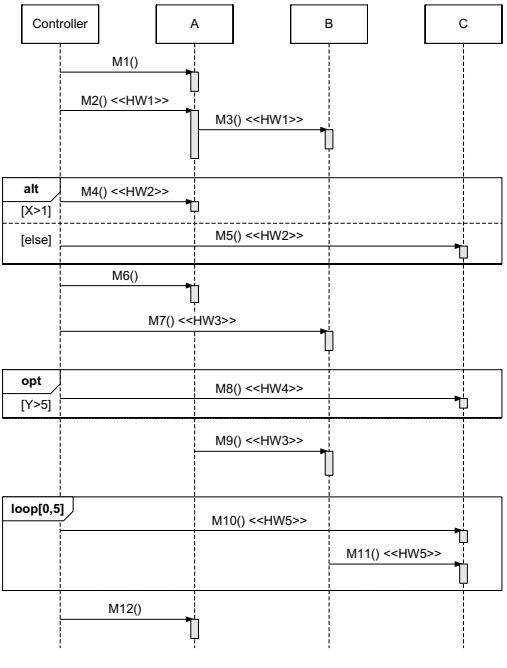


Fig. 3. The Example of the Partitioning on UML 2.0 Sequence Diagram

the guards. For example, if the guard [**x>1**] is evaluated to true, then the **M4()** method will be called, otherwise the [**else**] guard will be evaluated to true and then the **M5()** method will be called. The **opt** operator is the **if** portion of the **alt**, that is the same as the **if** construct in the C language. The **loop** operator defines that an interaction fragment shall be repeated several times.

When doing partitioning on the sequence diagrams, a designer may add a UML stereotype <<HWx>> for a method to be implemented in hardware. For example, in Fig. 3 the methods M2() and M3() are marked by the same stereotype <<HW1>>, but the method M4() is marked by another stereotype <<HW2>>. As a consequence, the methods M2() and M3() will be synthesized into the same bitstream, but the method M4() will be synthesized into another bitstream. Calling a hardware method that is synthesized into a different bitstream will require the FPGA to be reconfigured, therefore additional time overhead will be incurred.

The key performance penalties in DRCS are the FPGA reconfiguration time and the communication time between the CPU and the FPGA. These overheads are mainly dependent on the hardware restrictions. However, we can reduce the number of FPGA reconfigurations, so that reconfiguration overhead is decreased.

To reduce the number of FPGA reconfigurations, we need to take the control flow and execution order into consideration when doing partitioning on sequence diagrams. Hence, in DBPSD the following partitioning guidelines are provided:

Guideline 1: *Add the same stereotype <<HWx>> to all computing-intensive methods.* For example, M1() <<HW1>>, M2() <<HW1>>, ..., M12() <<HW1>>. If synthesis is feasible, only one bitstream is produced, thus only one reconfiguration action is needed.

Guideline 2: *Add the same stereotype <<HWx>> to all dependent methods.* For example, M3() is invoked by M2().

Guideline 3: *For the **alt** operator, add the same stereotype <<HWx>> to all the computing-intensive methods in all condition branches.* If the synthesis of the stereotyped methods is not possible, then start moving the last conditional branch to another stereotype <<HWy>> first, until synthesis is successful.

Guideline 4: *For the **opt** operator, associate all of the methods inside this interaction fragment with a stereotype different from the methods outside this interaction fragment.*

Guideline 5: *For the **loop** operator, associate the same stereotype <<HWx>> to all the computing-intensive methods inside this interaction fragment.* If the synthesis of the stereotyped methods is not possible, then start moving the less computing-intensive method to another stereotype <<HWy>> first, until synthesis is successful.

4 Implementation Example

An implementation example of DES (Data Encryption Standard) encryption system is presented to prove the feasibility of the proposed design flow and

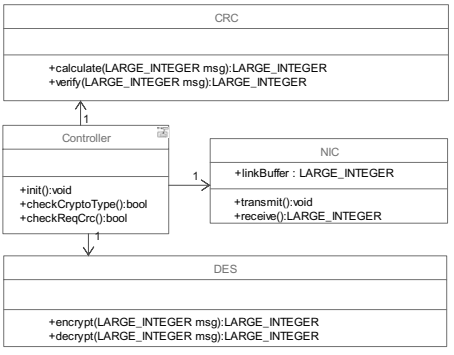


Fig. 4. The Class Diagram of the DES IMS System Example

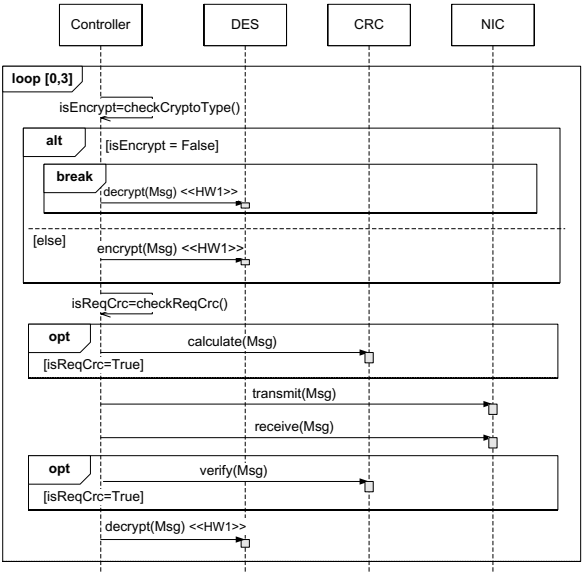


Fig. 5. The Sequence Diagram of the DES IMS System Example

Table 1. The Implementation Results of Different Partitions

	Partition 1	Partition 2	Partition 3	Partition 4
encrypt()	SW	SW	HW	HW
decrypt()	SW	HW	SW	HW
Total Execution Time	7200us	2560us	4880us	240us
FPGA Utilization (%LUTs)	0%	18%	18%	36%
Speedup Compared to Partition1	—	+2.81x	+1.48x	+30x

DBPSD partitioning methodology. Starting from the *design and implementation of the system software model* phase of the proposed design flow, the designer constructs the class diagram, the state machine diagrams, and the sequence diagrams for modeling this system, then the detailed functions are implemented in the C++ language.

Figure 4 is the class diagram for this system, which contains four classes: `Controller` (for controlling of the overall system), `NIC` (for simulating of the network interface), `CRC` (for calculating the CRC value), and `DES` (for encryption and decryption of message). Due to the limited space, the state machine diagram of this example is not shown.

The sequence diagram which depicts the overall system interaction is shown in Fig. 5. After the profiling procedure, we found that the `DES` class is a computing-intensive part of this system, thus the partitioning focuses on its two methods: encrypt() and decrypt(). Figure 5 shows the optimal partition for this example. The other partitions and their implementation results are shown in Table 1.

As shown in Table 1, the most notable partitions are Partition 2 and Partition 3. The difference in the total execution time of Partition 2 and Partition 3 was not expected. The reason for this difference can be observed from the sequence diagram of this example. Different control flows affect the number of times each method is invoked. Thus, the worth of doing partitioning on sequence diagram is proved by this example.

5 Conclusions

A UML-based design flow and its HW-SW partitioning methodology are presented in this article. The enhanced sequence diagram in UML 2.0 is capable of modelling complex control flows, thus the partitioning can be done efficiently on the sequence diagrams. As a result of using the proposed design flow, we are able to efficiently implement DRCS with significant reduction of error-prone, tedious, and time-consuming tasks, such as hardware design and HW-SW interface synthesis. Additionally, the real implementation results and information produced by the proposed flow such as application performance datum, hardware method execution time, FPGA reconfiguration time and communication overheads can used for further simulation or evaluation. Further research directions of this article include the semi-automatic or automatic HW-SW partitioning on sequence diagram, algorithm or methodology for reconfiguration overhead reduction, and support for FPGA partial reconfiguration.

References

1. K. Bondalapati and V. K. Prasanna, "Reconfigurable computing systems," Proceedings of the IEEE, 90(7):1201-1217, July 2002.
2. K. Compton and S. Hauck, "Reconfigurable computing: A survey of systems and software," ACM Computing Surveys, 34(2):171-210, June 2002.
3. G. Booch, J. Rumbaugh, and I. Jacobson, "Unified Modeling Language User Guide," Addison-Wesley, 1999.

4. I. A. Niaz and J. Tanaka, "Mapping UML statecharts to Java code," in Proceedings of the IASTED International Conference on Software Engineering (SE 2004), pages 111-116, February 2004.
5. Rhapsody case tool reference manual, I-Logix Inc, http://www.ilogix.com.
6. T. Beierlein, D. Frőhlich, and B. Steinbach, "UML-based co-design for run-time reconfigurable architectures," in Proceedings of the Forum on Specification and Design Languages (FDL03), pages 5-19, September 2003.
7. T. Beierlein, D. Frőhlich, and B. Steinbach, "Model-driven compilation of UML-models for reconfigurable architectures," in Proceedings of the Second RTAS Workshop on Model-Driven Embedded Systems (MoDES04), May 2004.
8. J. Fleischmann, K. Buchenrieder, and R. Kress, "A hardware/software prototyping environment for dynamically reconfigurable embedded systems," in Proceedings of the 6th International Workshop on Hardware/Software Codesign, pages 105-109, IEEE Computer Society, March 1998.
9. J. Fleischmann, K. Buchenrieder, and R. Kress, "Java driven codesign and prototyping of networked embedded systems," in Proceedings of the 36th ACM/IEEE Design Automation Conference (DAC99), pages 794-797, ACM Press, June 1999.
10. I. Robertson and J. Irvine, "A design flow for partially reconfigurable hardware," ACM Transactions on Embedded Computing Systems, 3(2):257-283, May 2004.

Hardware Task Scheduling and Placement in Operating Systems for Dynamically Reconfigurable SoC

Yuan-Hsiu Chen and Pao-Ann Hsiung

National Chung Cheng University, Chiayi, Taiwan–621, ROC
pahsiung@cs.ccu.edu.tw

Abstract. Existing operating systems can manage the execution of software tasks efficiently, however the manipulation of hardware tasks is very limited. In the research on the design and implementation of an embedded operating system that manages both software and hardware tasks in the same framework, two major issues are the dynamic scheduling and the dynamic placement of hardware tasks into a reconfigurable logic space in an SoC. The distinguishing criteria for good dynamic scheduling and placement methods include the total schedule length and the amount of fragmentation incurred while tasks are dynamically placed and replaced. Existing methods either do not take fragmentation into consideration or postpone the consideration of fragmentation to a later stage of space allocation. In our method, we try to reduce fragmentation during placement itself. The advantage of such an approach is that not only the reconfigurable space is utilized more efficiently, but the total schedule length is also reduced, that is, hardware tasks complete faster. Experimental results on large random tasks sets have shown that the proposed improvement is as much as 23.3% in total fragmentation and 2.0% in total schedule time.

Keywords: Operating System for Reconfigurable SoC, Hardware Scheduling, Placement, Dynamic Partial Reconfiguration.

1 Introduction

The advent of reconfigurable technologies in hardware design is making a significant impact on the design and the architectures of embedded systems and Systems-on-Chip (SoC). The hardware technologies for supporting reconfiguration are rapidly maturing, however the related software tools and design environments are relatively immature. For instance, we already have dynamic partial reconfiguration hardware such as Xilinx Virtex-II Pro FPGA; however the supporting software tools and runtime environments such as the support from embedded operating systems is still very much limited. In this work, we try to bridge this gap further by proposing scheduling and placement methods for hardware tasks in an SoC with dynamic partially reconfigurable logic.

In the design and implementation of an operating system for dynamically reconfigurable SoC, as shown in Fig. 1, hardware tasks can be dynamically scheduled and placed. For scheduling and placing hardware tasks, besides the desired efficiency, other important issues include the fragmentation of reconfigurable logic space and the total schedule time for a set of tasks. Existing methods either do not consider the issue of

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 489–498, 2005.
© IFIP International Federation for Information Processing 2005

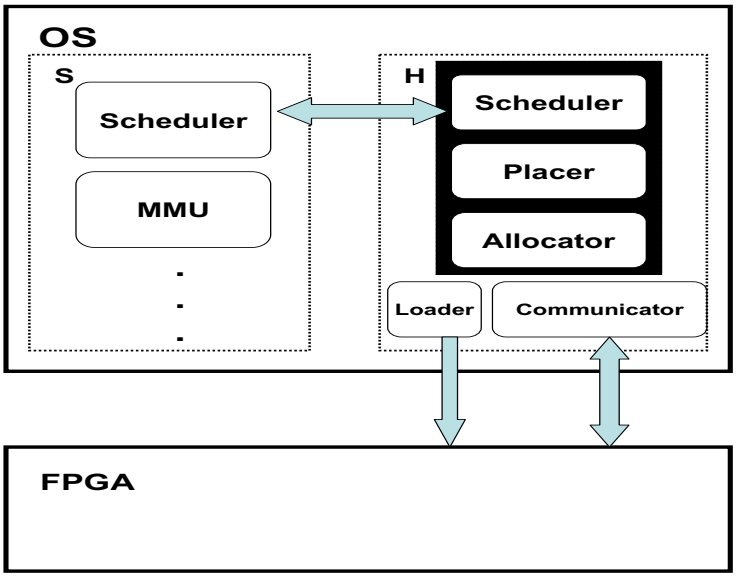


Fig. 1. Operating system for dynamically reconfigurable systems

fragmentation or postpone it to some later stage of space allocation. In this work, we will try to consider the fragmentation of reconfigurable space during scheduling and placement. The goal here is to reduce the amount of unused space that is wasted. As a result not only is the fragmentation reduced but the overall response and total schedule length of the tasks are also reduced.

FPGA is the most widely used reconfigurable logic nowadays. We will model the FPGA space and also the hardware tasks so that they can be used in the proposed scheduling and placement methods. Through experiments, we will also show how much reduction we obtained in fragmentation and in total time through a very simple task classification scheme that nearly does not incur any overhead during dynamic scheduling and placement. The techniques developed in this work will thus be very useful in the design of an operating system for dynamically reconfigurable systems.

The rest of this article is organized as follows. Section 2 presents some previous work in scheduling and placement of hardware tasks for dynamically reconfigurable systems. Section 3 describes the models for FPGA and for hardware tasks, which are used in our proposed method. Section 4 presents the two different scheduling algorithms that we used for scheduling the hardware tasks. Section 5 describes the proposed placement method for hardware tasks. Experimental results are given in Section 6. The article is concluded with future research directions in Section 7.

2 Previous Work

Scheduling hardware tasks is almost similar to scheduling software tasks. Scheduling methods can be classified into static and dynamic depending on when the scheduling

is done. Static scheduling can spend more time to find an optimal schedule, but it is fixed before run-time and thus cannot be changed once a system is running, making it quite inflexible. In contrast, dynamic scheduling must make scheduling decisions quickly so that the FPGA is not kept idle for too long without executing any hardware tasks; however the schedules might not be optimal. For static scheduling, variants of list scheduling have been proposed and used in several works, which differ mainly in their assignment of task priorities. Random priority list scheduling was used in the reconfigurable environment scheduling model [3] and dynamic priority list scheduling was proposed in [4].

For dynamic scheduling, the following methods have been proposed or used. Priority-based scheduling was used in the task and context schedulers [5]. Non-preemptive methods such as First-Come First-Serve (FCFS) and Shortest-Job First (SJF) scheduling were used in the online scheduler of [7]. Preemptive methods such as Shortest-Remaining Processing Time (SRPT) and Earliest Deadline First (EDF) were used also used in [7]. Steiger et al proposed a horizon and a stuffing method for dynamic scheduling and placement of hardware tasks [6].

The FPGA space has been modeled differently in the previous works. For example, in [5], the FPGA space was divided into slots that could accommodate only specific types of dynamic reconfigurable logics. In [7], the FPGA space was divided into slots of several fixed sizes and the placer selected the most suitably sized block for a task at hand, thus resulting in a smaller fragmentation. In [6], the FPGA space is considered as either a 1-dimensional or 2-dimensional area that can configured during scheduling and placement, for which two methods were proposed, namely horizon and stuffing.

Because our goal is to perform scheduling and placement of hardware tasks in an operating system, we focus on dynamic scheduling and placement methods. After a thorough survey we found that the 1-dimensional stuffing method [6] approximately meets our needs; however there are some issues such as significant fragmentation and schedule length so we decided to improve on this method such that we can decrease the resulting fragmentation and reduce the total execution time when hardware tasks are dynamically scheduled and placed in a reconfigurable SoC.

3 FPGA and Task Modeling

The FPGA configurable space is considered to be a set of columns, where a column spans the height of a chip and the set of columns spread across the width of the chip. The basic unit of configuration in this model is a column, which itself is a set of CLBs. This model fits the current technology of partial dynamically reconfigurable Xilinx Virtex II Pro FPGA chips. In this model, the reconfiguration is a 1-dimensional problem.

The hardware to be configured on an FPGA is modeled as a set of hardware tasks, where each task t has a set of attributes including arrival time $A(t)$, execution time $E(t)$, deadline $D(t)$, and area $C(t)$ (in terms of the number of FPGA columns required). These numbers can be obtained by synthesizing a hardware function using a synthesis tool such as Synplify or XST.

Given a set of hardware tasks and an FPGA space, our target problem is to schedule and place the tasks in the FPGA space such that the task attributes are all satisfied and

the time length of the schedule and the amount of FPGA space wasted (unconfigured) are minimized. As a solution to this problem, we proposed a *classified stuffing* method which improves on the original stuffing technique [6]. The scheduling and placement techniques in classified stuffing will be discussed in Sections 4 and 5, respectively.

4 Hardware Task Scheduling

The hardware tasks to be configured on FPGA may either have deadlines or not, according to which, we use different scheduling algorithms. For tasks without deadlines we use the *Shortest Remaining Processing Time* (SRPT) algorithm [1]. For tasks with deadlines we use the *Least Laxity First* (LLF) algorithm [2].

4.1 Hardware Scheduling Without Deadline

SRPT [1] is an optimal algorithm for minimizing mean response time and it has also been analyzed that the common misconception of unfairness in SRPT to large tasks is unfounded. For hardware tasks without deadline, we thus employ SRPT scheduling algorithm. The task which needs the least execution time will be assigned the highest priority. In the scheduler there is a queue to place hardware tasks that are ready for execution. The scheduler will insert the tasks into the ready queue according to their priorities.

4.2 Hardware Scheduling with Deadline

For scheduling real-time hardware tasks with deadlines, we can use either EDF or LLF. However, because hardware tasks are non-preemptive and truly parallel, we decided to use LLF, which degrades a little more gracefully under overload than EDF, extends relatively well to scheduling multiple processors, which are similar to parallel hardware tasks, and approaches the schedulability of EDF for non-preemptive tasks.

In LLF, the priority $P(t)$ of a task t is assigned as given in Equation (1), where now is the current time. Intuitively, the priority is the *slack* time for a task.

$$P(t) = D(t) - E(t) - now \tag{1}$$

5 Hardware Task Placement

In the 1D FPGA model, the smallest placement unit is one column of CLBs. Hence, we just need to know how wide a task is, that is, the number of columns required. If we only consider the task width for placing, it is a 1-dimensional problem. However, to more effectively utilize FPGA space, we also consider the task execution time, which makes the hardware task placement a 2-dimensional problem.

The main difference between our classified stuffing and the original stuffing method [6] is the classification of hardware tasks. In our method, we classify all hardware tasks into two types and the placement location of these two types of tasks are different, while in the stuffing method there is no such distinction. An example is shown in Fig. 2.

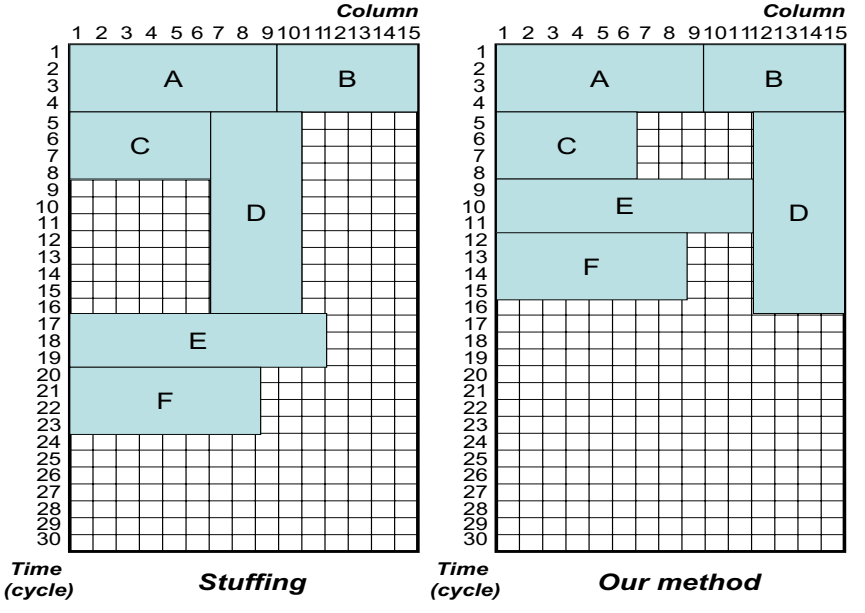


Fig. 2. The difference between our method and stuffing

Suppose tasks A, B, and C are already placed as shown in Fig. 2 and the placements are the same in our method and in stuffing. Next, task D is to be placed, however the location it is placed will be different in our method and in stuffing. In stuffing, it is placed adjacent to task C in the center columns of the FPGA space. In contrast, in our method task D will be placed from the rightmost columns. As a consequence, in our method tasks E and F can be placed earlier than that in stuffing, resulting in a shorter schedule. Finally, the fragmentation in our classified stuffing will also be lesser than that in stuffing because the space is used more compactly.

5.1 Task Classification

For a task t, we call the ratio $C(t)/E(t)$ the *Space Utilization Rate* (SUR) of t. Given a set of hardware tasks, they are classified into two types, namely *high SUR* tasks with $SUR(t) > 1$ and *low SUR* tasks with $SUR(t) \leq 1$. The classification of a task determines where it will be placed in the FPGA as follows. High SUR tasks are placed starting from the leftmost available columns of the FPGA space, while low SUR tasks are placed starting from the rightmost available columns. This choice of segregating the tasks and their placements is to reduce the conflicts between the high and low SUR tasks. A conflict often results in unused FPGA space, which in turn increases the total execution time. Because the classification scheme does not take the number of tasks of each type into account, one may wonder if this classification will not gain much when there is an imbalance between the number of high and low SUR tasks. However, after thorough experiments, we found the best results (least fragmentation, shortest execution time) are

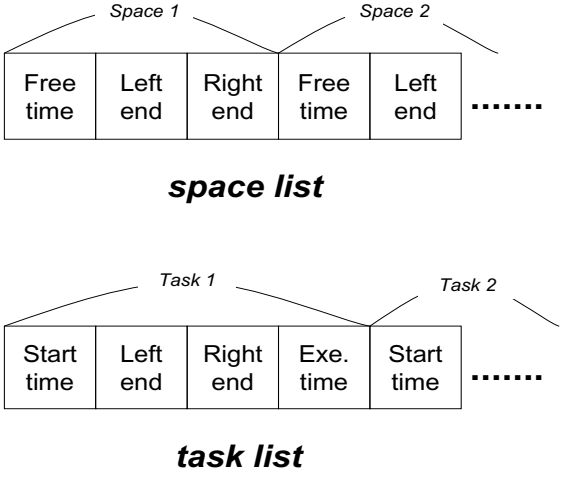


Fig. 3. Data structures used for recording placement and scheduling information

always obtained when we divide the task set considering only their SUR values. The results are described in Section 6 and Table 2.

5.2 Recording Placement Information

During placement, two lists are maintained, namely a *space list* for recording the free spaces and a *task list* for recording the task locations in FPGA after placement. For every free space in the space list we record three data including release time, the leftmost and the rightmost columns of this free space in FPGA. When the placer places a task into FPGA, the task list will store four information about the task, namely (1) the starting execution time of the task, (2) the leftmost column occupied by the task, (3) the rightmost column occupied by the task, and (4) the execution time of the task. The space list supplies information for the placer to find a fit free space for tasks, and the task list allows the placer to detect whether there is any conflict among the tasks. Figure 3 illustrates the two data structures. How to find a fit space and how to detect placement conflicts will be discussed in Section 5.3.

5.3 Placement Method

In the proposed classified stuffing method, placement of hardware tasks in FPGA space is performed as described in the following steps.

1. *Select a task:* The placer chooses a highest priority task from the ready queue according to our scheduling policy as described in Section 4. The task is classified into either high or low SUR as described in Section 5.1.
2. *Find a fit space:* Based on the width of a task, the placer will find a best fit free space from the space list. The best fit space is the earliest released of all free spaces which have enough columns of CLBs for the task. If there are more than one space

released at the same time, the leftmost space will be selected for placing a high SUR task, while the rightmost space will be selected for a low SUR task.

3. *Decide placement location in a free space:* At the second step, the placer found a best fit free space for a task, but this space maybe wider than that required by the task. In this case, according to the task type we determine which columns of CLBs of the best fit space will be used. A *high SUR* task will be placed from the leftmost column and a *low SUR* task will be placed from the rightmost column.
4. *Checking conflict:* At this step we already know where a task is to be placed, but we need to check if there is any conflict between the newly placed task and an existing already placed task. If a conflict exists, we must choose another fit space for the new task by going back to Step (2).
5. *Modify space list:* After a task is placed into a best fit space, not only the information of this fit space need to be modified, but also some other free spaces in the space list may be affected and thus need to be updated.

6 Experiment Results

To evaluate the advantages of the classified stuffing method compared with the original stuffing technique [6], we experimented with 150 sets of randomly generated hardware tasks, one-third of the sets had 50 tasks in each set, one-third had 200 tasks in each set, and one-third had 500 tasks in each set. The input task sets were also characterized by different *set ratios*, where the set ratio of a task set is defined as $|S_{>1}| : |S_{\leq 1}|$, where $S_{>1} = \{t \mid SUR(t) > 1\}$, $S_{\leq 1} = \{t \mid SUR(t) \leq 1\}$ so as to find out the kind of task sets for which our classified stuffing works better than stuffing. We experimented with three sets of tasks with set ratios 4:1, 1:1, and 1:4. The experiments were conducted on a P4 1.6 GHz PC with 512 MB RAM.

In the two-dimensional area of FPGA space versus execution time as illustrated in Fig. 2 each unit of placement is a *column-cycle*, where the space unit is a column and the time unit is a cycle. Table 1 shows the results of applying our method and the stuffing method to the same set of tasks, where the total column-cycles is the amount of column-cycles used by all tasks, the total fragmentation is the number of column-cycle not utilized, the total time is the total number of cycles for executing all tasks, the average fragmentation is the quotient of total fragmentation by total time, and the number of rejected tasks is the number of tasks whose deadlines could not be satisfied. For each task set, the data in Table 1 are the overall averages of the results obtained from experimenting with 50 different sets of randomly generated tasks.

From Table 1, we can observe that for both the methods and for all the sets of tasks the set ratio of 1:4 results in a shorter schedule and more compact placement as evidenced by the smaller fragmentation, which means that it is desirable to have hardware tasks synthesized into tasks with low SUR. The intuition here is that a greater number of high SUR tasks would require wider FPGA spaces resulting in larger fragmentation and thus longer schedules.

From Table 1, we can observe that for all the cases our method performs better than stuffing in terms of both lesser amount of fragmentation and lesser number of cycles. The total fragmentation is reduced by 5.5% to 23.3% and the total execution time is

Table 1. Experimental Results and Comparison

Task Set (S)			Original Stuffing				Classified Stuffing (reduction)				S&P Time (ms)
$\|S\|$	SR	TC	TF	TT	AF	TR	TF	TT	AF	TR	
50	4:1	1,711	505	105	4.8	3	477 (-5.5%)	104 (-1%)	4.6 (-4.2%)	3	3
	1:1	1,707	405	100	4	5	380 (-6.2%)	99 (-1%)	3.8 (-5%)	4	2
	1:4	1,550	418	93	4.5	2	392 (-6.2%)	92 (-1%)	4.3 (-4.4%)	2	2
200	4:1	6,780	1,237	381	3.25	3	1,119 (-9.5%)	376 (-1.3%)	2.97 (-8.6%)	3	5
	1:1	6,478	887	350	2.53	1	760 (-14.3%)	344 (-1.7%)	2.21 (-12.6%)	1	11
	1:4	6,133	823	330	2.49	0	739 (-10.2%)	326 (-1.2%)	2.27 (-8.8%)	0	12
500	4:1	17,145	2,246	922	2.43	5	2,007 (-10.6%)	911 (-1.2%)	2.20 (-9.5%)	5	47
	1:1	16,080	1,619	842	1.92	9	1,242 (-23.3%)	825 (-2.0%)	1.50 (-21.9%)	9	77
	1:4	15,077	1,341	782	1.71	5	1,185 (-11.6%)	775 (-0.9%)	1.53 (-10.5%)	5	98

SR: set ratio, TC: total column-cycles, TF: total fragmentation, TT: total time, AF: average fragmentation (TF/TT), TR: #tasks rejected, S&P: Scheduling and Placement

reduced by 0.9% to 2.0%. The reduction becomes more prominent in the case with set ratio 1:1 because there is an increased interference between the placements of the low SUR tasks and the high SUR tasks resulting in a larger fragmentation and longer schedules in the original stuffing method. These kind of interferences are diminished in classified stuffing. Comparing the task sets with different cardinalities, namely 50, 200, and 500, one can observe in Fig. 4 that the larger the task set is the better is the performance of our method. This shows that our method scales better than stuffing to more complex reconfigurable hardware systems.

As far as the performance of the scheduler and placer is concerned, the time taken by the classified stuffing method and the original stuffing method are almost the same. Thus, we have only one column for the S&P Time in milliseconds in Table 1. This is because the added classification technique does not also expend any significant amount of time, while it does improve the FPGA space utilization and total task execution time.

To support our choice of task classification using SUR value (see Section 5.1) and not considering the number of tasks of each type, we performed several experiments by using different classification ratios (CR) for a task set. The results are tabulated in Table 2, where we can observe that the best results are usually obtained when the classification

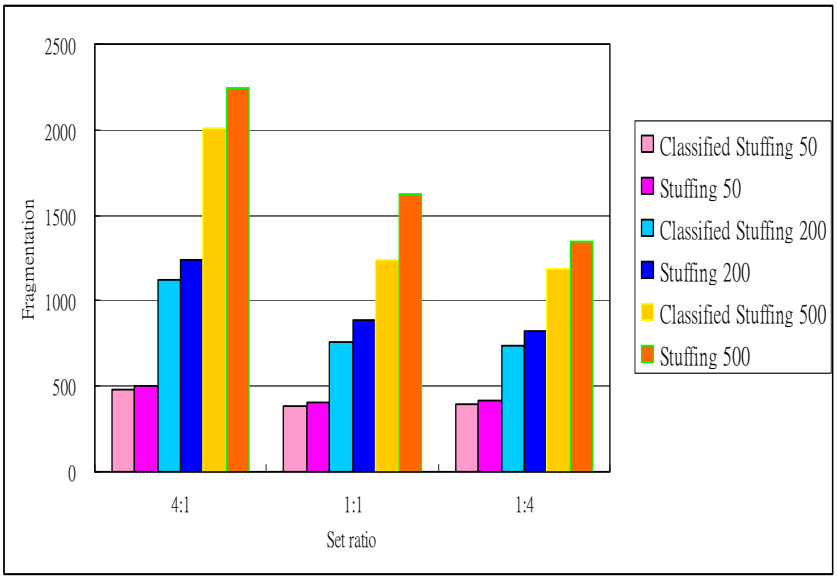


Fig. 4. Classified Stuffing vs. Original Stuffing

Table 2. Finding the Best Classification Ratio

Set Ratio	Scheduling Results	Classification Ratio		
		4:1	1:1	1:4
4:1	Total Fragmentation	**638**	667	667
	Total Time	**191**	192	192
1:1	Total Fragmentation	456	**435**	471
	Total Time	173	**171**	173
1:4	Total Fragmentation	458	438	**432**
	Ttotal Time	166	165	**165**

ratio is the same as the set ratio, that is, the threshold for classification can depend only on the SUR values.

7 Conclusions

A classified stuffing technique was proposed in this work showing significant benefits in both a shorter schedule and a more compact placement, but with little overhead. This was demonstrated through a large number of task sets. The current method focuses only on hardware scheduling. In the future, we will investigate hardware-software coscheduling.

References

1. N. Bansal and M. Harchol-Balter. Analysis of SRPT scheduling: investigating unfairness. In *Proc. of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 279–290. ACM Press, 2001.
2. J. Leung. A new algorithm for scheduling periodic, real-time tasks. *Algorithmica*, 4:209–219, 1989.
3. S.M. Loo and B. E. Wells. Task scheduling in a finite resource reconfigurable hardware/software co-design environment. *INFORMS Journal on Computing*, 2005. to appear.
4. B. Mei, P. Schaumont, and S. Vernalde. A hardware-software partitioning and scheduling algorithm for dynamically reconfigurable embedded systems. In *Proc. of the 11th ProRISC Workshop on Circuits, Systems, and Signal Processing*, 2000.
5. J. Noguera and R. M. Badia. System-level power-performance trade-offs in task scheduling for dynamically reconfigurable architectures. In *Proc. of the 2003 International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, pages 73–83. ACM Press, October 2003.
6. C. Steiger, H. Walder, and M. Platzner. Operating systems for reconfigurable embedded platforms: Online scheduling of real-time tasks. *IEEE Transactions on Computers*, 53(11):1392–1407, November 2004.
7. H. Walder and M. Platzner. Online scheduling for block-partitioned reconfigurable devices. In *Proc. of the Design Automation and Test, Europe (DATE)*, volume 1, pages 290–295, March 2003.

An Intelligent Agent for RFID-Based Home Network System

Woojin Lee, Juil Kim, and Kiwon Chong

Department of Computing, Soongsil University, Seoul, Korea
{bluewj, sespop}@empal.com, chong@comp.ssu.ac.kr

Abstract. An intelligent agent which is a software component for the efficient home network system is proposed in this paper. The agent consists of six modules such as the Agent Manager, the Data Collector, the Execution Controller, the Data Storage, the Data Queue and the User Interface. The Agent Manager manages the tasks of modules, and the Data Collector collects the data from home appliances through the RFID readers. The Execution Controller determines the operations of home appliances according to the conditions of the home environment and transfers the operations to the appliances through the RFID readers. Moreover, the Data Storage keeps the data which is necessary for the operations of the agent, and the Data Queue temporarily stores the data which is collected from home appliances. Lastly, the User Interface provides the graphical user interface in which an individual can directly control and monitor the home network. The proposed intelligent agent autonomously learns the circumstances of a home network by analyzing the data about the state of home appliances, and provides the best suited environment to the user. Therefore, the user can live in an optimal home environment without effort if he/she performs home networking through the agent.

1 Introduction

The PCs which are connected to the networks become important means for business support according as many PCs and networks connected with them are actively supplied. Moreover, not only wireless home networking which supports the communication between home appliances such as TVs, refrigerators, computers, PDAs and portable phones but also the unlimited communication between home appliances and the outside becomes possible because of the spread of the Internet and the recent rapid technological advances. Accordingly, home networking which binds home appliances is a hot issue in this century. Home networking means the digitalization of information; that is, information which can be processed is supplied to the home. This means that the efficiency of home life can be maximized through the home networking [1].

However, it is difficult to apply various home appliances in the current home network because IP addresses must be assigned to every appliance in the home network. It is too labor intensive to find proper appliances to communicate if a wireless LAN is used in the home network. Furthermore, a home network system must be modified in

* This work was supported by the Soongsil University Research Fund.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 499 – 508, 2005.
© IFIP International Federation for Information Processing 2005

case of the addition of new appliances; it is difficult to implement modules for TCP/IP communication.

To solve these issues, RFID (Radio Frequency IDentification) can be applied to the home network. RFID is an automatic identification technology that replaces traditional barcode technology. It reads and writes information without touching target objects using wireless communication technology. The barcode technology is developed and the price of a barcode is cheap, but they can be identified in close range without any obstruction between a scanner and a barcode because of the use of infrared rays. Moreover, it takes too long to identify a target; several of them cannot be identified at once. The amount of information stored is limited and the security is weak. RFID tags can be identified though there may be an obstruction between reader and tag because RFID uses radio frequency instead of infrared rays, and the tags can be identified at a maximum distance of 30 meters. Furthermore, several targets can be identified at the same time and many more kinds of information can be securely stored and communicated according to the change of time and conditions [2].

Various appliances can be incorporated into a home network because home networking is performed using radio frequency without assigning IP to each appliance if the RFID is applied to the home network. Moreover, a home network system need not be modified much when new appliances are added to the home network. More efficient home networking will be implemented because it is easy to implement and apply the RFID-based home network system.

Therefore, an intelligent agent for efficiently managing the RFID-based home network is stated here. The efficiency of home life at home will be maximized through the home network which has an intelligent agent that learns by itself.

2 Related Works

2.1 Home Network

A home network is a set of components that connects and integrates several appliances which perform calculation, management, monitoring and communication in order to process, manage, transmit and store information in a house. The home network is composed of association of equipments more than two which have the capability to share data and communicate. The structure of a home network is composed of a home gateway which connects the inside network with the outside network, middleware which controls the communication network and information devices in the house, and devices which include the functions for home networking. This is implemented through various networking protocol such as Ethernet, telephone lines, power lines and wireless. It also facilitates the sharing of functions and data, and the remote control between home appliances connected with the network. Furthermore, it offers Internet access, audio/video streaming, and home control applications and services [3].

2.2 RFID (Radio Frequency IDentification)

RFID is a means of storing and retrieving data through electromagnetic transmission to an RF compatible integrated circuit, and is now being seen as a radical means of

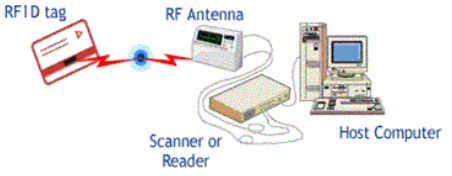


Fig. 1. Components of RFID System

enhancing data handling processes [4]. An RFID system has several basic components including a number of RFID readers, RFID tags, and the communication between them. The components of RFID system are represented in Figure 1.

The RFID reader can scan data emitted from RFID tags. RFID readers and tags use a defined radio frequency and protocol to transmit and receive data [5].

RFID tags are classified into three sub-classes, namely passive, semi-passive, and active. Passive RFID tags do not require batteries for operation and are, therefore, inherently robust, reliable, and low-cost. Their construction is relatively simple. A high performance passive RFID tag consists of a tiny integrated circuit chip, a printed antenna, and an adhesive label substrate for application to items. Active and semipassive tags require batteries for operation and, therefore, provide a greater range and throughput than passive (batteryless) tags. The simple addition of a battery to an RFID tag is a necessary but incomplete feature that classifies it as active [6][7].

2.3 RFID/USN (Ubiquitous Sensor Network)

RFID/USN manages information through the network by detecting real time information of objects which attach RFID. It binds itself to anything with not only its sensory information but also the information of its surroundings such as temperature, humidity, pollution and cracks. The ultimate purpose of the RFID/USN is to implement ubiquitous environments. RFID/USN focuses on the RFID which stores data, and sensing capability will be added to the RFID in the near future. Finally, networking capability will be added to the RFID [8][9].

The use of RFID/USN will increase step by step according to the development of sensor technology. RFID tags will become smaller in size and more intelligent while the price of them will decrease. Accordingly, the use of RFID will be expanded in the fields of logistics, distribution, circumstances, prevention of accidents, management of medical care and management of food. RFID will be developed into a smarter micro network sensor by adding sensing and communication facilities. In the future, the RFID/USN will be evolved into the level that can recognize and manage surroundings by multifunctional tags from the current level that recognizes the fixed code of objects, and will be developed into an intelligent USN which has a communication facility [9].

3 Architecture of RFID-Based Home Network

Home appliances with RFID sensors which are RFID tags with sensing facillity, RFID readers that read data from RFID tags and write data to RFID tags, and a home

server which controls the home network are necessary for the construction of a RFID-based home network. Moreover, mobile devices which can communicate with a home server using wireless LAN are needed in order to control the home network while a user is moving. Figure 2 is the structure of this RFID-based home network.

RFID sensors[7] are attached to all appliances in order to sense the state of appliances in the RFID-based home network. RFID sensors periodically assess the state of home appliances and write the state on to RFID tags. RFID readers read the data from RFID tags and transfer the data to the home server. The intelligent agent of the home server collects and analyzes the data from RFID readers, determines the operations of home appliances in order to maintain an optimal environment for the house owner, and controls the home appliances to perform the operations by transferring them through RFID readers. Furthermore, the agent provides the service that the house owner can directly control and monitor the home network.

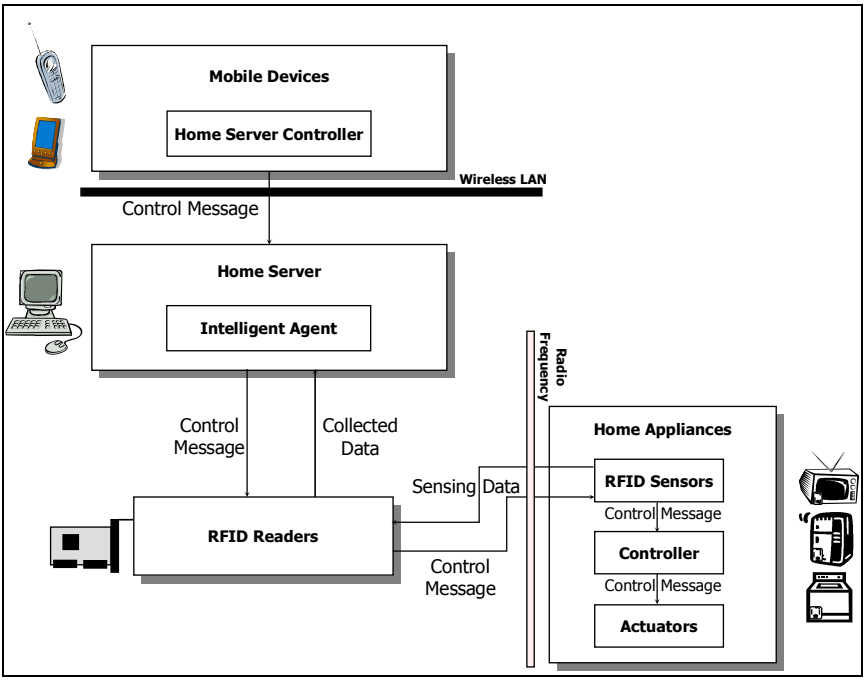


Fig. 2. Architecture of RFID-based Home Network

The following is the process of RFID-based home networking.

① RFID sensors periodically sense the state of home appliances. They sense the current state of appliances, store the state in tags, and transfer the state to RFID readers.
② RFID readers read the state of home appliances from RFID sensors. Not only do RFID readers receive the data transferred from RFID sensors, but also they periodically check the RFID sensors and read data about the state of home appliances.

③ The intelligent agent of the home server collects data from RFID readers, analyzes the data, and determines the operations of home appliances in order to provide an optimal environment to the user of home network.
④ The intelligent agent of the home server sends control messages to the RFID readers after it determines the operations.
⑤ RFID readers transfer the control messages received from the home server to home appliances.
⑥ Home appliances perform operations according to the control messages.

4 Intelligent Agent for RFID-Based Home Networking

An intelligent agent which autonomously controls RFID-based home networking is proposed in this section.

The user of a home network must periodically monitor the environment of his/her home network and manage the home network to maintain an optimal home environment for him/her. This is a labor intensive process on the part of the user. Accordingly, an agent that can autonomously control the home network instead of the user is necessary in order to save the effort and the time of the user. This agent must continuously collect the data from RFID sensors, analyze the collected data, and find out the inclinations of the user based on the analyzed information. Moreover, it must continuously monitor and control the home network to provide an optimal home environment according to the desires of the user.

4.1 Structure

The intelligent agent collects data from RFID sensors, analyzes the data, and controls the home network to maintain the most comfortable home environment based on the data. Figure 3 represents the intelligent agent for RFID-based home networking. This agent consists of six modules: the Agent Manager, the Data Collector, the Execution Controller, the Data Storage, the Data Queue and the User Interface.

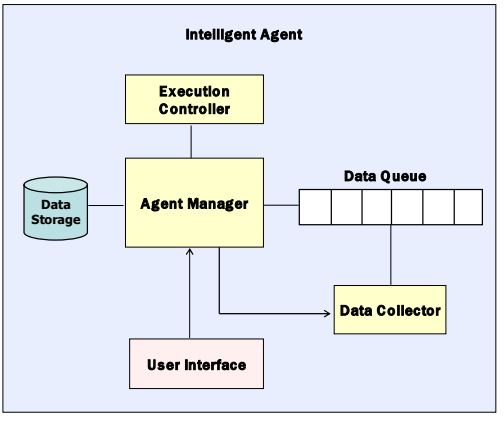


Fig. 3. Intelligent Agent

Agent Manager

The management of intelligent agent and study about the inclination of the user are necessary in order to maintain the best suited home networking for the user.

The Agent Manager manages modules in the agent so that they can accomplish accurately their tasks. In addition, it determines the best suited home networking for the house owner by analyzing the data which is gathered by the Data Collector and stored in the Data Queue.

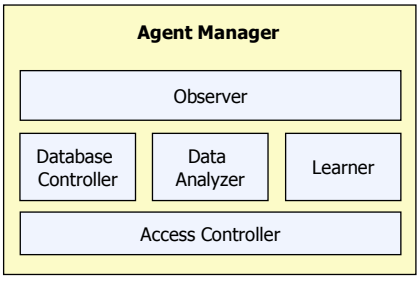


Fig. 4. Agent Manager

Figure 4 represents the structure of the Agent Manager. The Agent Manager consists of the Access Controller, the Database Controller, the Data Analyzer, the Learner and the Observer.

The following is the description of the five modules in the Agent Manager:

* Access Controller: This supervises the access to the agent from the outside. Only authorized users can access the agent in order to directly monitor and control the home network.
* Database Controller: This provides basic primitives for database operations, such as selection, insertion, update and deletion. The manipulation of information in the Data Storage is performed using this module.
* Data Analyzer: This analyzes the data read from RFID sensors. It looks at the data about the state of home appliances from RFID sensors which are attached to the home appliances in the home network and understands the current circumstances of the home network.
* Learner: This component studies the inclination of the user based on the information from the Data Analyzer. It imitates the wishes of the user from the past and current analyzed information about the environment of the home network. The following is the algorithm of the Learner. The Learner studies the inclination of the user based on the data of home appliances during the past 7 days. Every hour it analyzes the data of the same time zone during the past 7 days and understands the inclination of the user based on the repeated data over the setting probability.
* Observer: This monitors the execution of modules in the agent and manages the modules to correctly perform their roles.

```
Define a device which has just on/off information as SimpleDevice.
Define a device which has on/off and additional information as ComplexDevice.

begin
  for i = 0 to (SimpleDevice_Count)
    for j = 1 to 24  // get data of recent 7 days  from SimpleDevice_Tables
      Execute query "Select Count(*), action from SimpleDevice_Tables[i]
                     WHERE datetime = j and user = CurrentUser and date >= CurrentDate-7 group by action";
      count_on := count of on;
      count_off := count of off;
      total_count := count_on + count_off;
      if( count_on >= (total_count*probability) ) then  // probability is a user-defined percentage
        SimpleDeviceInfo[i][j] := on;                   // total_count *probability is frequency of the action
      else if (count_off >= (total_count*probability) ) then
                SimpleDeviceInfo[i][j] := off;
      end if;
    end for;
  end for;
  for i = 0 to (ComplexDevice_Count)
    for j = 1 to 24  // get data of recent 7 days  from ComplexDevice_Tables
      Execute query "Select Count(*), action from ComplexDevice_Tables[i]
                     WHERE date >= CurrentDate-7 and datetime = j and user = CurrentUser group by action";
      count_on := count of on
      count_off := count of off
      total_count := count_on + count_off;
      if( count_on >= (total_count*probability) ) then
        Execute query "Select Count(*), additionalInfo from ComplexDevice_Table[i]
                       WHERE date >= CurrentDate-7 and datetime = j and action = on and user = CurrentUser
                       group by additionalInfo";
        info := most frequent additionalInfo;
        ComplexDeviceInfo[i][j] := on;
        ComplexDeviceadditionalInfo[i][j] := info;
      else if( count_off >= (total_count*probability) ) then
        ComplexDeviceInfo[i][j] := off;
      end if;
    end for;
  end for;
end;
```

Data Collector

It is necessary that the states of home appliances are continuously read from RFID sensors in order to maintain the home environments of the user.

The Data Collector continuously sends messages to the RFID readers, collects new data received from RFID sensors, and stores the data in the Data Queue. It collects the data from the RFID readers if the RFID sensors sense the states of the home appliances and actively sends the data to the RFID readers. In addition, it sends messages to the RFID readers to read the data from the RFID sensors according to the necessity.

Execution Controller

The proper operations according to the conditions should be determined, and performed by the home appliances in order to maintain the best suited home environments for the user after studying about the inclination of the user.

The Execution Controller determines the operations of home appliances according to the conditions of the home environment in order to maintain an optimal home environment for the user. It also sends control messages to the RFID readers in order to control the home appliances.

Data Storage
The information such as the previous operations and the inclination of the user is necessary in order to autonomously maintain the home environments. For this, the storage which stores the information is necessary.

The Data Storage is the database of the agent. Each module of the agent gets the data for operations from the Data Storage, performs its operations and stores the data which must be preserved in the Data Storage according to the result of the operations. The information about the wishes of the user, which is received by analyzing the data of the home network, is stored in the Data Storage and it is updated continuously.

Data Queue
The data which is periodically read from RFID sensors should be temporarily stored in order to analyze the current home environments.

The Data Queue is temporary storage. The data which is read from the RFID sensors is held here for a short time before it is processed. RFID sensors which are attached to home appliances periodically transfer the data to the RFID readers simultaneously and the Data Collector gathers the data which is transferred from several RFID sensors at a time. Therefore, it is necessary to temporarily store the data so that the Execution Controller can analyze the data and perform proper operations.

User Interface
It should be easy and convenient that the user directly controls the home network.

The User Interface is a GUI (Graphical User Interface). The intelligent agent autonomously maintains the optimal home network environment according to the desires of the user, but the user can control the home network using this interface if he or she wants. Moreover, the user can monitor the home network using this interface.

4.2 Collaborations

The modules of the intelligent agent collaborate with each other as figure 5 in order to control the home network. This collaboration shows that the agent autonomously controls the home network. The user can perform these tasks through the User Interface if he or she wants to directly control the home network.

* The Data Collector collects the data that the RFID sensors attached to home appliances such as an indoor thermometer, refrigerator, TV, lighting and water heater, sense the states of the appliances and send to the RFID readers, and stores the collected data in the Data Queue. For example, the Data Collector gathers the data such as the current indoor temperature, the current temperature of refrigerator and the on/off state of the TV, and stores the data in the Data Queue.
* The Agent Manager reads the collected data from the Data Queue, analyzes the data, learns the wishes of the user of the home network and stores the results in Data Storage. For example, the Agent Manager reads the data about the indoor temperature, air conditioner and water heater, and analyzes the data. As a result of the analysis, the Agent Manager may find out that the user of the home network turned on the air conditioner if the indoor temperature was higher than 25°C and turned on the water heater if the indoor temperature was lower than 20°C. The Agent Manager learns that the user prefers indoor temperature between 20°C and 25°C, and stores the result in the Data Storage.

* The Execution Controller determines the operations of home appliances according to the current circumstances based on the desires of the user. For example, the Execution Controller may determine that the air conditioner should be turned on in order to decrease the indoor temperature if the current indoor temperature is higher than 25°C because the user prefers an indoor temperature between 20°C and 25°C.
* The Execution Controller stores the determination in the Data Storage and transfers it to the home appliances through RFID readers to perform the operations. For example, the Execution Controller stores the indoor temperature and the operation of the air conditioner in the Data Storage if it determines that the air conditioner should be turned on, and transfers the information to the RFID sensor attached to the air conditioner. Then the air conditioner is turned on through its controller.

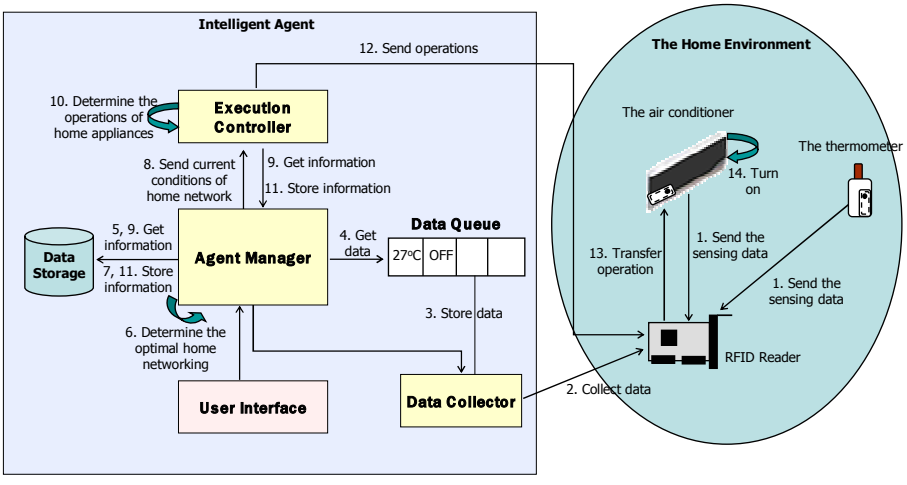


Fig. 5. Collaborations of Modules in the Intelligent Agent

5 Conclusion and Future Work

An intelligent agent for the efficient management of the home network is proposed in this paper. The agent consists of six modules: the Agent Manager, the Data Collector, the Execution Controller, the Data Storage, the Data Queue and the User Interface.

The Agent Manager manages the tasks of the modules in the agent, and the Data Collector collects the data from the home appliances through the RFID readers. The Execution Controller determines the operations of the home appliances according to the conditions of the home network and transfers them to the home appliances through the RFID readers. Moreover, the Data Storage keeps the data which is necessary for the operations of the agent, and the Data Queue temporarily stores the data which is collected from the home appliances. Lastly, the User Interface provides the

graphical user interface that the user of the home network can directly control and monitor the home network.

The proposed intelligent agent autonomously learns the circumstances of the home network by analyzing the data about the states of home appliances, and provides the best suited environment for the user. Therefore, the user can live in the most comfortable home environment without effort if he or she performs home networking through the agent.

The proposed intelligent agent will be developed in the near future. Furthermore, a mobile intelligent agent which can control home network that home appliances communicate with each other through RFID will be studied.

References

[1] Changhwan Kim, *The Technical Trend of Wireless Home Networking*, Electronics Information Center of Korea Electronics Technology Institute, 2004.

[2] Junghwan Kim, Cheonkyo Park and Yonggyun Kim, *The Development Direction and Introduction Guideline of RFID*, Institute of Information Technology Assessment, 2004.

[3] Yuncheol Lee, *Recent Technical Trend and Market View of Home Networking*, Electronics and Telecommunications Research Institute, 2003.

[4] Mario Chiesa, Ryan Genz, Franziska Heubler, Kim Mingo, Chris Noessel, Natasha Sopieva, Dave Slocombe, Jason Tester, *RFID*, http://people.interaction-ivrea.it/c.noessel/RFID/research.html, 2002.

[5] Lionel M. Ni, Yunhao Liu, Yiu Cho Lau and Abhishek P. Patil, *LANDMARC: Indoor Location Sensing Using Active RFID*, Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003.

[6] Bridgelall, R., *Bluetooth/802.11 Protocol Adaptation for RFID Tags*, Proceedings of the 4th European Wireless Conference, 2002.

[7] Bridgelall, R., *Enabling Mobile Commerce through Pervasive Communications with Ubiquitous RF Tags*, 2003 Wireless Communications and Networking, 2003.

[8] Ministry of Information and Communication Republic of Korea, *The Basic Plan of U-Sensor Network Construction*, 2004.

[9] Seunghwa Yu, *The Propulsive Direction of RFID/USN Standardization*, The Journal of TTA, No. 94, 2004, pp12-18.

An Intelligent Adaptation System Based on a Self-growing Engine

Jehwan Oh, Seunghwa Lee, and Eunseok Lee

School of Information and Communication Engineering, Sungkyunkwan University,
300 Chunchun jangahn Suwon, 440-746, Korea
{hide7674, jbmania, eslee}@selab.skku.ac.kr

Abstract. In this paper, a self-growing engine based adaptation system, which automatically decides the more efficiency plan about the assigning of jobs, in a mobile, grid computing environment, is proposed. Recently, research relating to grid computing has become an important issue, achieving certain goals by sharing the idle resources of computing devices, and overcoming various constraints of the mobile computing environment. In this domain, most existing research assigns work only by considering the status of resources. Hence the situation of assigning work to a peer having relatively low work efficiency, is possible. The proposed system considers various contexts and selects the most suitable peer. In addition, the system stores the history of the work result, and if the same request occurs in the future, a peer is selected by analyzing the history. In this paper, a prototype used to evaluate the proposed system is implemented, and the effectiveness of the system is confirmed through two experiments.

1 Introduction

In modern society, the handheld device is quickly becoming widespread, its usage and explosive growth can be attributed to the rapid development of wireless network technologies. The computing power of these handheld devices is continually increasing, relative to the growth of associated technologies. However, handheld devices have limited capacity because of a potable feature. For this reason, applications for handheld devices to some extent, are limited, and developers must create applications offering limited services. The expectations of users are increasing, demanding the same the quality of service as the desktop computer. Grid computing technologies, creating distributed processing, operating over a number of low capacity computers, can perform large amounts of work, which is difficult to process in an individual computing device. This technology overcomes the various constraints of handheld devices, and creates large scale applications over a wireless environment.

In grid computing, the allocation of work to the various devices is a significant problem. Most existing research adopts a method of allocating the work only by considering the resource status of the participating peers. Therefore, problems could arise of assigning work to peers, which have relatively low work efficiency. For instance, the processing time of a peer which has the byte code for work, even if the peer has insufficient resources, may still be faster than the processing time of a peer which has sufficient resources, but does not have the byte code.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 509 – 517, 2005.
© IFIP International Federation for Information Processing 2005

To cope with this problem, the proposed system selects the most suitable peer through considering current variable context (resource usage pattern, available resource, network bandwidth, existence of the byte code), therefore more efficient distribution of work is made possible. In addition, the proposed system monitors the processing results for work and stores these values in the history DB. Then, the system computes the average values for each factor and retrieves the more significant context factor. The system assigns the weight value to the selected factor, selects the peer with the most similar context values, and average values among the current participating peers, assigning the work to the selected peer. Next, the results better than average values are stored in the history DB. The system has self learning feature, selecting the more suitable host according to the time passed.

In this paper, a prototype is implemented. In order to evaluate the proposed system, the effectiveness of the system is confirmed through two experiments.

The paper is organized as follows: Section 2 introduces related work. Section 3 describes the comprehensive structure and the behavior process of the proposed system. In Section 4, details of the operational process and algorithm are presented. System evaluation through the implementation of a prototype is described in Section 5. Finally, Section 6 concludes the paper and identifies future work.

2 Related Work

Research overcoming the various constraints has been studied in various laboratories. Firstly, research relating to adaptation, which is adjusting the quality of service, and parameters of the application or system components, has been actively processed [1][2][3][4]. However, this method focuses on the quality of service offered in the past.

Grid computing is becoming an important issue, achieving various goals by sharing idle resources of diverse computing devices. The system can perform large amounts of work, which is difficult to process for individual computing devices. Representative studies include Globus [7] and Condor [8]. However, these studies focus on the wired environment and therefore are not suitable for the mobile environment. Studies dealing with the problems of the mobile environment include the following; Xiaohui Gu *et al.* [6] proposes the Adaptive offloading System, performing distributed processing in a surrogate computer, having relatively rich memory to overcome the limited memory available in mobile devices. Another study, Junseok Hwang *et al.*[10]proposes middleware based on a mobile proxy structure that can execute jobs submitted to mobile devices, in-effect making a grid consisting of mobile devices. However, these systems only consider the available resources of each peer when allocating the jobs. Thus, the possibility of work being assigned to a peer with relatively low work efficiency, exists.

In this paper, an intelligent system is proposed, this system is based on resource usage patterns, existence of the byte code, as well as available resources, and has a self learning feature.

In the subsequent section, the proposed system applying these concepts, is described in detail.

3 Proposed System

3.1 System Framework

This paper proposes the intelligent adaptation system, which continually improves the method of allocating jobs, by using a self growing engine, in order to efficiently allocate jobs. As presented in Fig. 1, the central server monitors/stores the context information of peers. If peer A requests services from the server, then the server selects the peers which can perform the requested services according to various computation processing, allocating the jobs to selected peers. In addition, the system stores the history of the work result, and if the identical request is issued in the future, it selects the most suitable peer by analyzing the history.

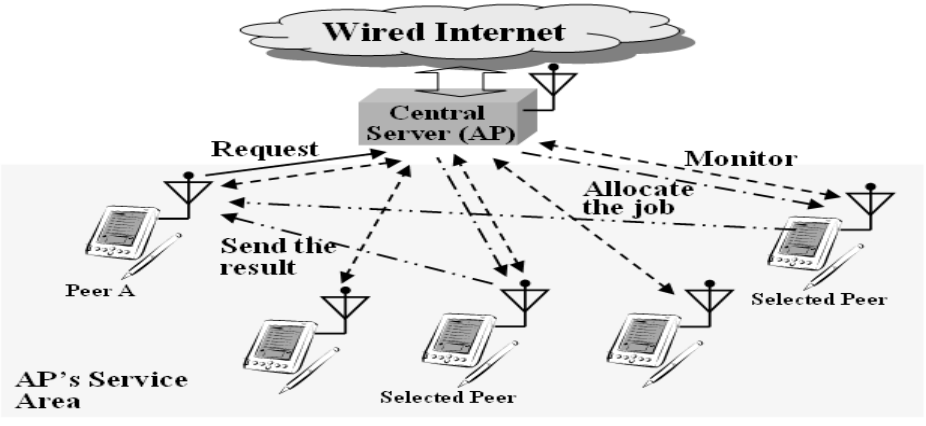


Fig. 1. Overall Architecture of Proposed System

3.2 System Components

As presented in Fig.2 the component of the proposed system is composed of two parts; the Client Module (CM), which is embedded in each peer, and the Server Module (SM), which is embedded in the central server. The SM is located at or close to a wireless network access point. The role of each principal component is as follows.

- *Context Observer (CO):* gathers the variety context information including resource usage pattern and transmits this information to the *SM*
- *Task Division Agent (TDA):* divides the self-executing jobs , entrusted depending on gathered context
- *Integration Agent (IA)*: integrates the received job results from the external agent, and delivers the integrated results to the application
- *Task Executor (TE)*: executes the jobs required from an external peer
- *Communicator (client)*: performs the interaction with the *SM* or peers
- *Communicator (sever)*: performs the interaction with the *CM* in each peer

- *Analysis Agent (AA):* selects the job history DB corresponding to the requested job from the client from the selected job history DB, and computes the average value of a client's condition values when a client performs the each job. In addition, the *AA* stores the context information received from each client, in the Client Information DB. The AA then analyzes the results of the executed job by the selected peer, and updates the job history DB with the result.
- *Decision Agent (DA):* decides peers to be allocated based on various computations and on the peer's condition analyzed by the *AA*
- *Client Information DB*: The location where various context information (resource usage pattern, available resource, network bandwidth, existence of the byte code) of each participating peer is stored. This information is periodically updated.
- *Job history DB*: The location where the executed result each service is stored.

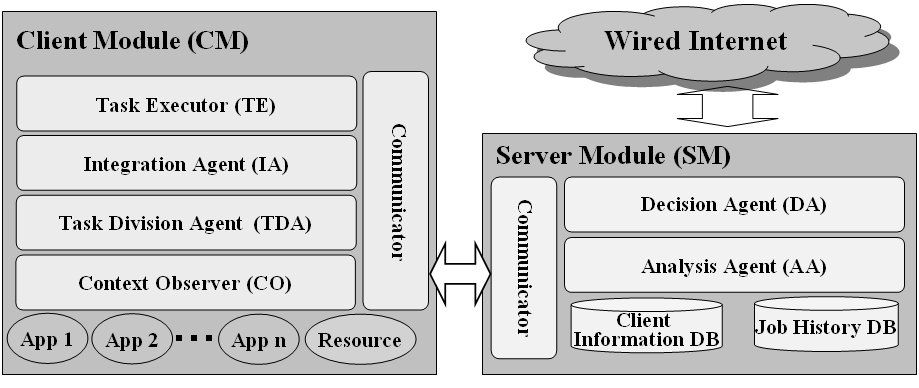


Fig. 2. Components of the Proposed System

3.3 Overall System Behavior

A sequence diagram corresponding to the overall system behavior is presented in Fig.3.

Firstly, when a user connects to the server, the CO continually gathers the current resource status including the resource usage pattern, and transmits this information to the SM. Then, the AA receives this information, and computes the score for this context, storing this score value in the Client Information DB.

Next, when a user executes the application, the TDA decides the jobs to be executed at the current resource status. The jobs, which cannot be executed, are entrusted to the AA via a communicator. The AA retrieves the suitable condition of the peer to execute the Jobs in the Job History DB. The DA finds the peer based-on the retrieved condition of peer AA in Client Information DB. The selected peer is allocated to the jobs. The TE of the selected peer performs the jobs, and then transmits the executed results to the CM requesting the jobs. The IA of the CM receives and integrates the results. Finally, the IA transmits the integrated result to the application.

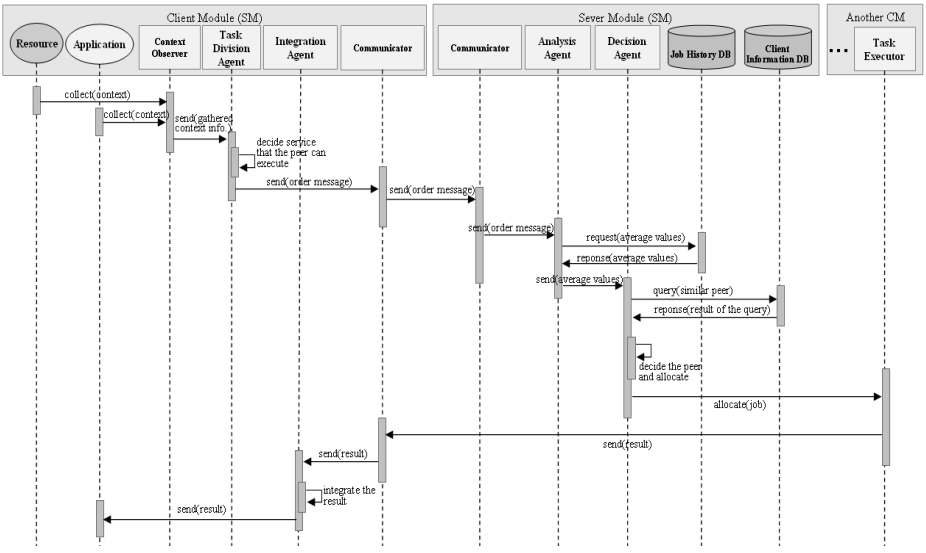


Fig. 3. Sequence Diagram of Overall System Behavior

3.4 Selection Phase of the Peer Via Computation

In the case of the server allocating the requested jobs from the client only by considering the resource status of each participating peer, the case of assigning work to a peer which has a relatively low work efficiency, could exist. For instance, the processing time of a peer which has the byte code for the work may have insufficient resources, but achieves results faster than the processing time of a peer having sufficient resources, but does not have the byte code. To cope with these problems, the proposed system selects the most suitable peer through considering more relevant current context as follows:

- The current available (idle) resources of each participating peers such as CPU, RAM, Battery.
- The network bandwidth of each participating peer
- The changing probability of resources computed via the resource usage pattern analysis of each participating peer
- The existence of the byte code, which will be the completed job

This information is collected in the CM, and periodically stored in the Client Information DB. Then each factor is computed to a relative score using the formulas presented in Table 1.

The SM can obtain the final score of each device using a formula as follows. Then, the SM selects the device, which obtained the highest final score, allocating the jobs to the selected device. The each formula is differently created to become range of values from 0.01 to 1.00.

$$CPU_{level} \bullet w_1 + RAM_{level} \bullet w_2 + \cdots = Final\,Score \tag{1}$$

Table 1. The Sample Formulas to Compute the Each Factor

Context type	Formula	Example	Score
Idle resource	$1-\frac{CPU_{available\ Rate}\times CPU_{clock}}{10000}=CPU_{level}$	$1-\frac{20\times 400}{10000}=0.2$	2
	$\frac{RAM_{freeSpace}}{RAM_{size}}=RAM_{level}$	$\frac{20}{64}=0.312$	4
	:	:	:
The Changing Probability of Resource(w)	$1-\frac{\sqrt{\frac{(x_1-m)^2+(x_2-m)^2+\cdots+(x_n-m)^2}{n}}}{10}$	$1-\frac{2.49}{10}=0.75$	8
:	:	:	:

```
(REQUEST
   :sender(agent-identifier :name CM1@HIDE:1099/)
   :receiver (set(agent-identifier :name SM@selab:1099/))
   :context "(action translator)and
             (action image_compression)
        ...
)
```

```
(INFORM
   :sender(agent-identifier :name CM1@HIDE:1099/)
   :receiver (set(agent-identifier :name SM@selab:1099/))
   :context "(service_name(translator)),(entrusted_device(tester76)),
             (execution_time(3.5)),
        ...
)
```

Fig. 4. Example of ACL message, which is requesting the jobs from the CM to the SM, informing the completion of work from the CM to the SM

3.5 Learning Phase Via Self Growing Engine

The selecting phase of the specific peer is described via computation, in the previous section, when the CM requests the jobs from the SM. In this section, the self learning of the most suitable peer using the history for the job result, is introduced.

When the peer requests a job, the AA retrieves the average value of the performed results in the Job History DB. The AA transmits the retrieving result to the DA. The DA assigns the weight to the element having the highest value among the average values, and finds the peer, which is similar to this value in the Client Information DB. If a similar peer exists, the SM allocates the jobs to the similar peer. However, if a similar peer does not exist, the DA selects the most suitable peer via computation, described in the previous section, and the selected peer is allocated the jobs. The TE of the selected CM performs the jobs, then transmits the executed results to the CM requesting the jobs. The CM receiving the result informs the time for executing work and the condition of the device performing the jobs to the AA. The AA analyzes the received information from the CM. If the processing time is less than the average

time, the AA stores this information in the job history DB. The AA re-computes the average value in the job history DB.

Table 2 presents sample result data for the translator service in the Job history DB. The average values mean that the condition is the suitable condition of the peer executing the translator service.

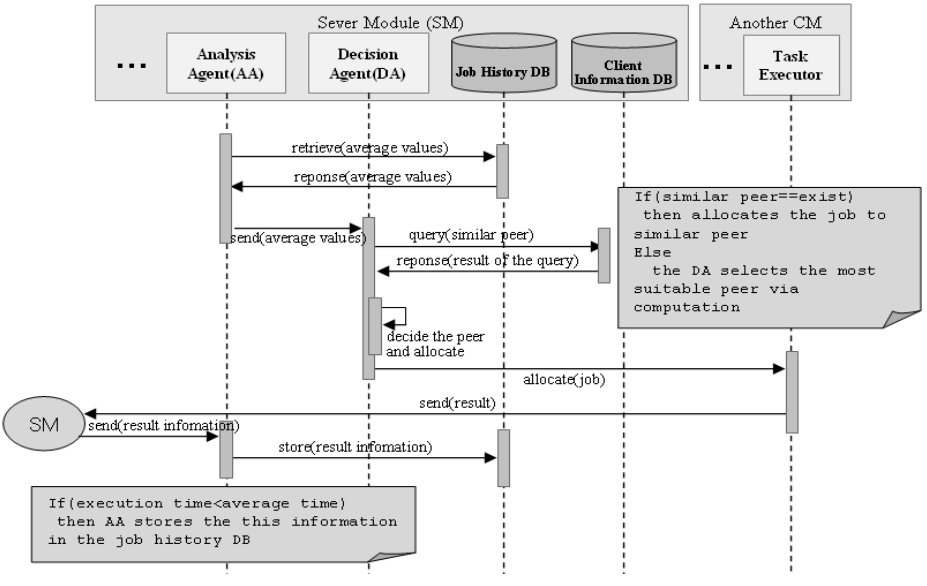


Fig. 5. Sequence Diagram of Learning Phase

Table 2. Sample of the Result Data for the Translator Service in the Job history DB

Device name	Condition	Execution time	Date
Tester 34	7, 5, 4, 1, 4, 8	3.23	20050704
Tester 21	4, 4, 5, 2, 3, 4	3.74	20050621
Tester 28	3, 9, 4, 1, 4, 7	3.59	20050520
:	:	:	:
Average value	**4.7, 7.3, 6.1, 1.4, 7.2, 3.4**	**3.82**	

4 System Evaluation

Each prototype module for the system evaluation was implemented mainly using Embedded Visual C++. In this experiment, the CM and SM are both implemented on the desktop PC, connected by wired Internet. The CM, is tested using a PDA simulator offered by the Embedded Visual Studio. The executed application is 'Remote Video Medical Treatment' for the E-healthcare scenario, studied in the laboratory. The divided jobs are 'Translator', 'Image Converter', and 'Video Converter' respectively.

The efficiency of the proposed system is confirmed via two experiments. Firstly, the time for processing jobs is compared by considering the resource status, with the time for processing the jobs. The information of 10 peers is stored in the Client Information DB, and requests generating the translator service are generated in a peer.

Consequently, the proposed system selects the peer using factors such as network bandwidth, existence of job byte code, and available resources. The processing time of the peer selected by the proposed system is superior to the processing time of a peer selected when only considering the resource status. The result for this experiment is presented in Table 3.

Table 3. Comparison of Processing Time

	Condition of Selected Peer	Processing Time (s)
The existing method	6, 8, 9	5.6
The proposed method	4, 7, 8, 1, 8, 9	4.3

Secondly, a change in time for processing the job according to the number of times that process the same job increases, is observed. The translator service is executed 100 times using the PDA simulator, and the average of the processing times and the conditions of the PDA simulators is computed. Initially the average values are stored in the Job history DB. The peer repeatedly requests the translator service from the SM.

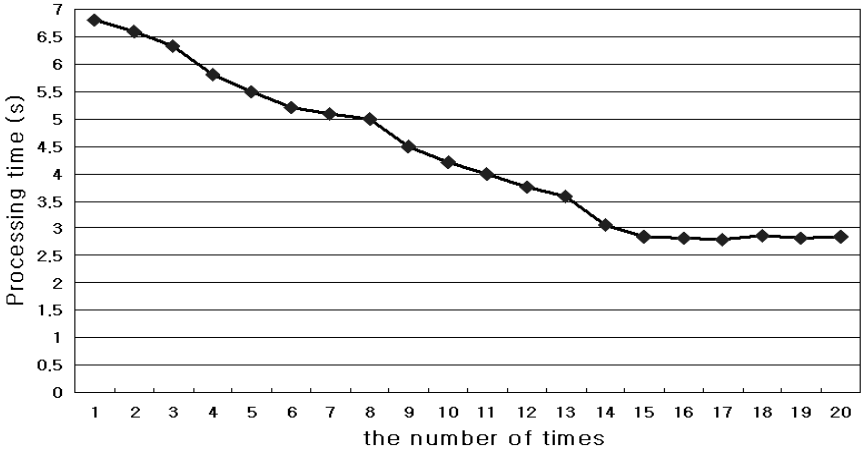


Fig. 6. Decrease of the processing time through a repeat execution of the translator service

Fig. 6 presents the experiment results. When the peer initially requests the translator service, the processing time is similar to the average processing time. Accordingly, as the number of times that the peer executes the translator service increases, the processing time decreases. However, when the peer executes the translator service 15 times, the processing time does not decrease. This result means that the SGE finds the most efficient condition for executing the translator service.

It is confirmed that the proposed system is more efficient than the traditional method of allocating jobs.

5 Conclusion

In this paper, a self learning engine based adaptation system to more efficiently plan and improve the current system, in the mobile grid computing environment, is presented. The efficiency of the proposed system is confirmed via two experiments. The various aspects of the proposed system are expected to solve the constraints discussed in wireless computing, resulting in a more convenient wireless computing environment.

Additionally we see a number of other key future research areas. First is the investiga-tion of more diverse context that may occur in mobile environment. Second is study for more efficient algorithms for gathering and analyzing context. Last is study for more optimal size of history data.

References

1. Alvin T.S. Chan, Siu-Nam Chuang, "MobiPADS: A Reflective Middleware for Context-Aware Mobile Computing", IEEE Transaction on Software Engineering. Vol. 29, No.12 pp.1072-1085, Dec.2003.
2. Brian Noble, "System Support for Mobile, Adaptive Applications", IEEE Personal Communications Vol.7 No.1, Feb.2000
3. Wai Yip Lum, Francis C. M. Lau, "User-Centric Content Negotiation for Effective Adaptation Service in Mobile Computing", IEEE Transaction on Software Engineering. Vol. 29, No.12 pp.1100-1111, Dec.2003
4. Vahe Poladian, João Pedro Sousa, David Garlan, Mary Shaw, "Dynamic Configuration of Resource-Aware Services", 26th International Conference on Software Engineering (ICSE'04), pp.604-613, May.2004
5. Richard S. Sutton, Andrew G. Barto, 'Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)', The MIT Press, Mar.1998
6. Xiaohui Gu, Klara Nahrstedt, Alan Messer, Ira Greenberg, Dejan Milojicic, "Adaptive offloading for pervasive computing", Pervasive Computing, IEEE Volume 3, Issue 3, pp.66-73, July-Sept. 2004
7. http://www.globus.org/
8. http://www.cs.wisc.edu/condor/
9. Gabrielle Allen, David Angulo, Ian Foster, Gerd Lanfermann, Chuang Liu, Thomas Radke, Ed Seidel and John Shalf, "The Cactus Worm: Experiments with Dynamic Resource Discovery and Allocation in a Grid Environment", Journal of High-Performance Computing Applications, Volume 15, no. 4 2001
10. Junseok Hwang; Aravamudham, P., "Middleware services for P2P computing in wireless grid networks", Internet Computing, IEEE, Volume 8, Issue 4, pp.40-46, July-Aug. 2004

Dynamically Selecting Distribution Strategies for Web Documents According to Access Pattern*

Wenyu Qu[1], Di Wu[2], Keqiu Li[3], and Hong Shen[1]

[1] Graduate School of Information Science,
Japan Advanced Institute of Science and Technology,
1-1, Asahidai, Nomi, Ishikawa, 923-1292, Japan
[2] Department of Computer Science and Engineering,
Dalian University of Technology,
No 2, Linggong Road, Ganjingzi District, Dalian, 116024, China
[3] College of Computer Science and Technology,
Dalian Maritime University,
No 1, Linghai Road, Dalian, 116026, China
keqiu_01@163.com

Abstract. Web caching and replication are efficient techniques for reducing web traffic, user access latency, and server load. In this paper we present a group-based method for dynamically selecting distribution strategies for web documents according to access patterns. The documents are divided into groups according to access patterns and the documents in each group are assigned to the same distribution strategy. Our group-based model combines performance metrics with the different weights assigned to each of them. We use both trace data and statistical data to simulate our methods. The experimental results show that our group-based method for document distribution strategy selection can improve several performance metrics, while keeping others almost the same.

Keywords: Web caching and replication, distribution strategy, cache replacement algorithm, simulation, trace data, autonomous system (AS).

1 Introduction

In recent years, the effective distribution and maintenance of stored information has become a major concern for Internet users, as the Internet becomes increasingly congested and popular web sites suffer from overloaded conditions caused by large numbers of simultaneous accesses. When users retrieve web documents from the Internet, they often experience considerable latency.

Web caching and replication are two important approaches for enhancing the efficient delivery of web contents, reducing latencies experienced by users. A user's request for a document is directed to a nearby copy, not to the original server, thus reducing access time, average server load, and overall network traffic. Caching [4] was originally applied to distributed file systems. Although

* The preliminary version of this paper appeared in [8]. Corresponding author: K. Li.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 518–527, 2005.
© IFIP International Federation for Information Processing 2005

it has been well studied, its application on the Internet gave rise to new problems, such as where to place a cache, how to make sure that cached contents are valid, how to solve replacement problems, how to handle dynamic web documents, etc. Replication was commonly applied to distributed file systems to increase availability and fault tolerance [9]. Both techniques have complementary roles in the web environment. Caching attempts to store the most commonly accessed objects as close to the clients as possible, while replication distributes a site's contents across multiple replica servers. Caching directly targets minimizing download delays, by assuming that retrieving the required object from the cache incurs less latency than getting it from the web server. Replication, on the other hand, accounts for improved end-to-end responsiveness by allowing clients to perform downloads from their closest replica server.

Although web caching and replication can enhance the delivery efficiency of web contents and reduce response time, they also bring some problems, such as maintaining consistency of documents, propagating content updates to replica servers and caches, and so on. There are many ways to distribute copies of a web document across multiple servers. One has to decide how many copies are needed, where and when to create them, and how to keep them consistent. A good distribution strategy would be an algorithm that makes these decisions. We argue that there is no distribution strategy that is optimal for all performance metrics; in most cases, we have to pay the cost of making some performance metrics worse if we hope to make one or more of the others better. In this paper we present a group-based method for dynamically selecting distribution strategies for web documents according to access patterns. We divide the documents into groups according to access patterns and assign the same distribution strategy to the documents in each group. Further, we present a group-based model that combines performance metrics with the different weights assigned to each of them. Therefore, our method can generate a family of strategy arrangements that can be adapted to different network characteristics. To realize our method, we use a system model [8] in which documents can be placed on multiple Internet hosts . Clients are grouped based on the autonomous systems (ASs) that host them. ASs are used to achieve efficient world-wide routing of IP Packets [3]. In this model, each AS groups a set of clients that are relatively close to each other in a network-topological sense. In this paper, we consider a more general system model, in which an intermediate server is configured either as a replica server, or a cache, or neither. Finally, we use both trace data and statistical data to simulate our methods. The experimental results show that our group-based method for document distribution strategy selection can outperform the global strategy and improve several performance metrics compared to the document-based method, while keeping the others almost the same.

The rest of the paper is organized as follows. Section 2 focuses on our group-based method for dynamically selecting distribution strategies for web documents according to access patterns. The simulation experiments are described in Section 3. Finally, we conclude our paper in Section 4.

2 Selection of Document Distribution Strategy

In this section, we first briefly outline the distribution strategies used in this paper, and then we present a group-based method for dynamically selecting distribution strategies for web documents according to access patterns.

2.1 Distribution Strategies

We considered the following document distribution strategies.

1. No Replication ($NoRepl$): This is a basic strategy that does not use any replication at all. All clients connect to the primary server directly.
2. Verification (CV): When a cache hit occurs, the cache systematically checks the copy's consistency by sending an If-Modified-Since request to the primary server before sending the document to the client. After the primary server revalidates the request, the intermediate decides how to get the document for the client.
3. Limited verification (CLV): When a copy is created, it is given a time-to-live (TTL) that is proportional to the time elapsed since its last modification. Before the expiration of the TTL, the cache manages requests without any consistency checks and sends the copies directly to the client. After the TTL expires, the copies are removed from the cache. In our experiments, we used the following formula to determine the TTL. $\alpha = 0.2$ is the default in the Squid cache [3].

$$T_r = (1 + \alpha)T_c - T_l \tag{1}$$

where T_r is the expiration time, T_c is the cached time, and T_l is the last modified time. α is a parameter which can be selected by the user.
4. Delayed verification (CDV): This policy is almost identical to the CLV strategy, in that a copy is created, it is also given a TTL. However, when the TTL expires, the copies are not removed from the cache immediately; the cache sends an If-Modified-Since request to the primary server before sending the copies to the client. After the primary server revalidates the request, the intermediate decides how to fetch the document for the client.

Ideally, we would have as many replica servers as ASs, so every client could fetch the needed document from the replica server; this, in turn, would produce good results on some performance metrics such as hit ratio and byte hit ratio. However, on the other hand, it also would make other performance metrics, such as consumed bandwidth and server load, worse.
5. $SU50$ (Server Update): The primary server maintains copies at the 50 most relevant intermediate servers.
6. $SU50 + CLV$: The primary server maintains copies at the 50 most relevant intermediate servers; the other intermediate servers follow the CLV strategy.

2.2 A Group-Based Method for Document Distribution Selection

First we introduce a method to group the documents into P groups according to their access patterns. The main factors that influence the access patterns are web

resource and user behavior. According to [7], we group the documents according to the value of v_d, which is defined as follows:

$$v_d = (c_d + f_d/u_d)s_d \tag{2}$$

where c_d denotes the cost of fetching document d from the server, f_d denotes the access frequency of document d, u_d denotes the update frequency of document d, and s_d denotes the size of document d. We can see that when P is equal to the number of the documents, i.e., when there is only one document in each group, then our method is the same as the document-based method in [11]. Therefore, from this point of view the method proposed in [11] can be viewed as a special case of our method. For the case of $P = 1$, our method can be considered a global strategy method, since all the documents are assigned to the same strategy.

Now we present our group-based model considering the total effect of the performance metrics from a general point of view, e.g. we define the total function for each performance metric according to its characteristics. The existing method [11] defines the total function for each performance metric by summing the performance metrics of each document. We argue that this method does not always work well for some performance metrics such as total hit ratio.

Let $S = \{s_j, j = 1, 2, \cdots, |S|\}$ be the set of distribution strategies, $G = \{G_j, j = 1, 2, \cdots, |G|\}$ be the set of groups, $M = \{m_j, j = 1, 2, \cdots, |M|\}$ be the set of performance metrics such as total turnaround time, hit ratio, total consumed bandwidth, etc. A pair arrangement $(strategy, group)$ means that a strategy is assigned to the documents in a group. We denote the set of all the possible arrangements as A. We can define a function f_k for each metric m_k on a pair $a \in A$ by $R_{ka} = \sum_{j=1}^{|G|} r_{kaj} = \sum_{j=1}^{|G|} f_k(a, G_j)$, where R_{ka} is the aggregated performance result in metric m_k and r_{kaj} is the performance result in metric m_k for G_j .

Let $w = \{w_1, w_2, \cdots, w_{|M|}\}$ be the weight vector which satisfies:

$$\sum_{k=1}^{M} w_k = 1, w_k \geq 0, k = 1, 2, \cdots, |M| \tag{3}$$

We can get the following general model $R_a^* = \min_k w_k R_{ka}$. We refer to R_a^* as the total cost function for different weight vector w for an arrangement a .

Since there are a total of $|S|^{|G|}$ different arrangements, it is not computationally feasible to achieve the optimal arrangements by the brute-force assignment method. The following result shows that it requires at most$|G||S|$ computations to obtain an optimal strategy arrangement for the documents in each group.

$$\begin{aligned} R_a^* &= \min_{a \in A} \sum_k w_k R_{ka} = \min_{a \in A} \sum_{k=1}^{|M|} w_k (\sum_{j=1}^{|G|} r_{kaj}) = \min_{a \in A} \sum_{k=1}^{|M|} \sum_{j=1}^{|G|} w_k r_{kaj} \\ &= \min_{a \in A} \sum_{j=1}^{|G|} \sum_{k=1}^{|M|} w_k r_{kaj} \geq \min_{a \in A} \sum_{j=1}^{|G|} (\min_j \sum_{k=1}^{|M|} w_k R_{kaj}) \end{aligned}$$

From the above reasoning, we can obtain the total optimal arrangement by computing the optimal arrangement for each group. Therefore, the computation is the sum of that for obtaining the optimal arrangement for the documents in each group, whereas the computation workload for the method in [11] is about $|D||S|$, where $|D|$ is the total number of documents. Thus, our method requires less computation than the method in [11] by $(|D| - |G|)|S|$. If we suppose that there are 100 documents, and we divide the documents into 10 groups, we can see that the computation can be reduced by 90%.

In our experiments we mainly considered the following performance metrics: (1) Average Response Time per request (ART): the average time for satisfying a request. (2) Total Network Bandwidth (TNB): the total additional time it takes to transfer actual content, expressed in bytes per milli-second. (3) Hit Ratio (HR): the ratio of the requests satisfied from the caches over the total requests. (4) Byte Hit Ratio (BTR): the ratio of the number of bytes satisfied from the caches over the total number of bytes.

For the case of $k = 1, 2$, suppose $max = \max_j R_{kj}$, $min = \min_j R_{kj}$. Before defining the total performance metric result function for the case of $k = 1, 2$, we should apply a transformation $f(R_{kj}) = (R_{kj} - min)/(max - min)$ on R_{kj} so that $f(R_{kj}) \in [0, 1]$. Therefore all the performance metric results are in the interval $[0, 1]$. Otherwise it is not feasible to decide the weights for the performance metrics. For example, in the case of $ART = 150$, $TNB = 200$, $HR = 0.9$, and $BHR = 0.9$, let $w = (0.45, 0.45, 0.05, 0.05)$. In this case, we can see that HR and BHR play little role in the total cost, although the weights of them are very large. For ART and TNB, we define $R_k = \sum_{j=1}^{|D|} f(R_{kj})/|D|, k = 3, 4$.

For $HR(k = 3)$, let $R_{1j} = f_1(s_i, G_j)$ be the number of requests that hit in the replica servers and the caches for the pair (s_i, G_j). We define $R_1 = \sum_{j=1}^{|D|} R_{1j}/NR$, where NR is the total number of requests.

For $BHR(k = 4)$, let $R_{2j} = f_2(s_i, G_j)$ be the number of bytes that hit in the replica servers and the caches for the pair (s_i, G_j). We define $R_2 = \sum_{j=1}^{|D|} R_{2j}/NBR$, where NBR is the total number of requests bytes.

3 Simulation

In this section we use trace data and statistical data to simulate the methods proposed in previous sections. In the simulation model, we assume that the primary server has the privilege of updating the documents whose copies are distributed or stored in the replica servers and the caches. A replica server always holds the document; a cache may or may not hold it. In the following figures, "Per-Group" and "Per-Document" represent the performance results of our group-based method and the existing document-based method.

3.1 Simulation with Trace Data

In this section we apply trace data to simulate our results. The trace-based simulation method is similar to that introduced in [10]. In our experiments, we collected traces from two web servers created by the Vrije Universiteit Amsterdam in the Netherlands (VUA) and the National Laboratory for Applied Network Research (NLANR). Table 1 shows the general statistical data for the traces.

Table 1. Statistics of Trace Data

Issue	VUA	NLANR
Start Date	September 19, 1999	March 27, 2001
End Date	December 24, 1999	April 11, 2001
Duration (days)	96	16
Number of Documents	26,556	187,356
Number of Requests	1,484,356	3,037,625
Number of Creates	26,556	187,356
Number of Updates	85,327	703,945
Number of ASs	7,563	90

Table 2. Performance Results for Per-Group Strategy

$w = (w_1, w_2)$	VUA		NLANR	
	TNB(GB)	ART(Sec)	NB(GB)	TT(hours)
(0.9,0.1)	95.3	8.82	162.2	5.37
(0.8,0.2)	110.2	6.95	175.7	5.68
(0.7,0.3)	126.5	6.24	196.7	5.83
(0.6,0.4)	136.5	5.86	212.5	6.04
(0.5,0.5)	150.7	5.57	256.5	6.27
(0.4,0.6)	167.4	5.33	283.5	6.62
(0.3,0.7)	178.2	5.20	314.5	6.89
(0.2,0.8)	191.7	5.11	346.8	7.05
(0.1,0.9)	205.6	5.05.1	379.4	7.24

In this section we describe our experiment for assigning the same distribution strategy to the documents in each group. The simulation results shown in Table 2 were obtained when the number of groups was 100 and 200 for VUA and NLANR, respectively. We simulated a case in which there are two performance metrics, ART and TNB.

From Figure 1 we can see that the results of our method approximate those of the existing method when we group the documents into 117 and 211 groups for VUA and NLANR, respectively. From our experiments, we conclude that there is almost no improvement in result as the number of groups increases. However, our method can significantly improve both the procedure execution time and the memory management cost, as can be seen in Figures 2 and 3.

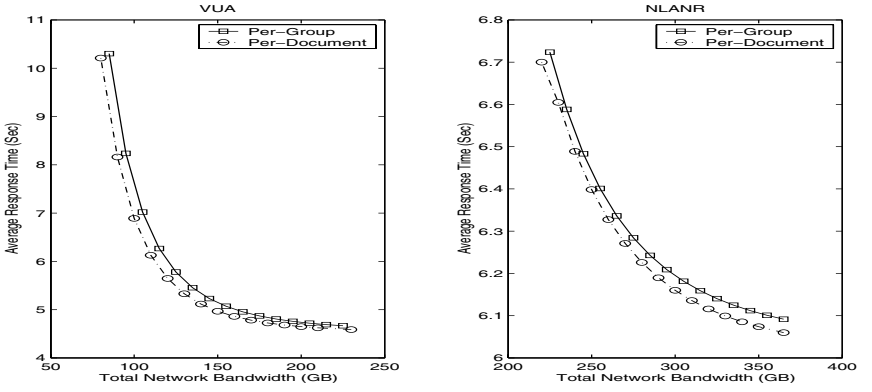


Fig. 1. Different Arrangements

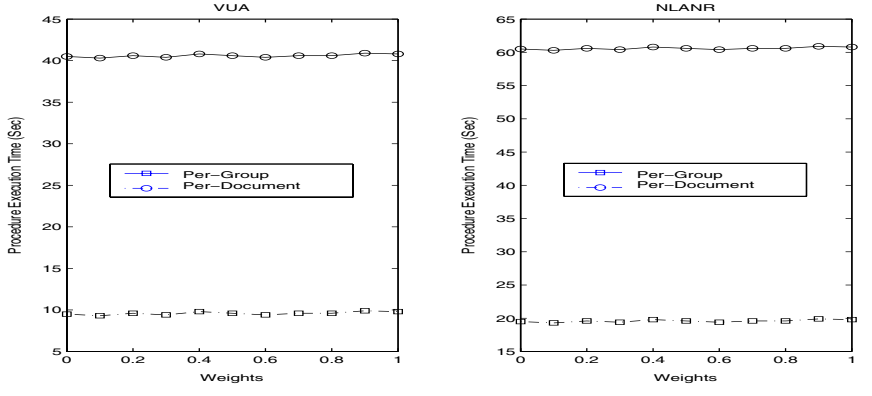


Fig. 2. Procedure Execution Time

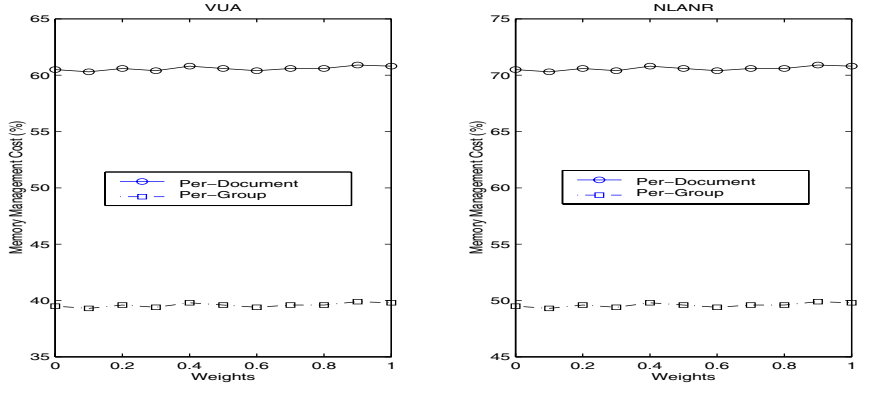


Fig. 3. Memory Management Cost

3.2 Simulation with Statistical Data

In this section we use statistical data to simulate our methods. The parameters shown in Table 3 are chosen from the open literature and are considered to be reasonable [1, 2, 5, 6]. We have conducted experiments for many topologies

Table 3. Parameters Used in Simulation

Parameter	Value
Number of Nodes	200
Number of Web Objects	5000
Number of Requests	500000
Number of Updates	10000
Web Object Size Distribution	Pareto Distribution $p(x) = \frac{ab^a}{a-1}$ $(a = 1.1, b = 8596)$
Web Object Access Frequency	Zipf-Like Distribution $\frac{1}{i^\alpha}$ $(i = 0.7)$
Delay of Links	Exponential Distribution $p(x) = \theta^{-1}e^{-x/\theta}$ $(\theta = 0.06\ Sec)$
Average Request Rate Per Node	$U(1, 9)$ requests per second

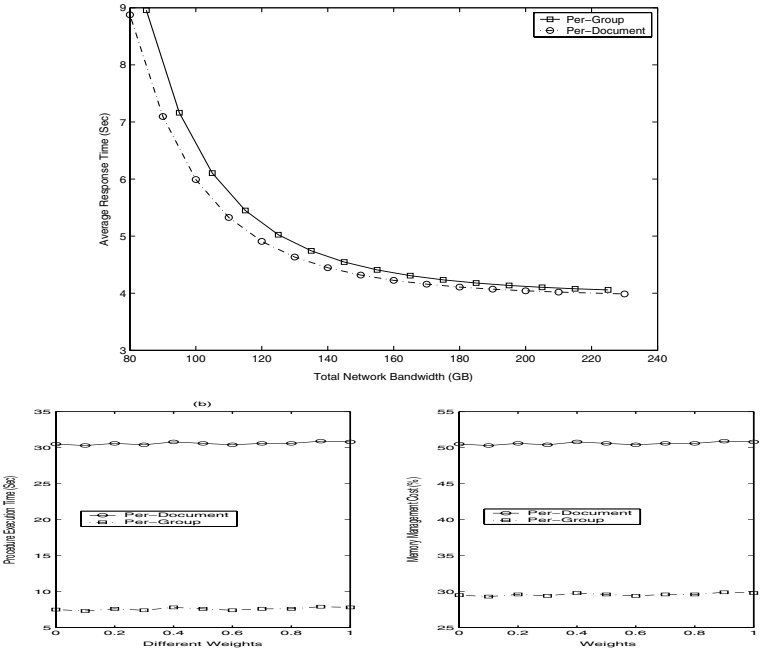


Fig. 4. Performance Results for Per-Group Strategy

with different parameters and the performance of our results was found to be insensitive to topology changes. Here, we list only the experimental results for one topology, due to space limitations.

From Figure 4 we can see that the results of our method approximate those of the existing method when we group the documents into 89 groups. However, our method can improve both the procedure execution time and the memory management cost.

4 Concluding Remarks

Since web caching and replication are efficient ways to reduce web traffic and latency for users, more and more researchers have been paying a lot of attention to this topic. In this paper, we presented a method for dynamically selecting web replication strategies according to the access patterns. We also used both web trace and statistical data to simulate our method. However, we can see that there will be performance problems when more strategies are considered. In the future, this work should be extended to the replication of other types of objects, since we considered only static objects in this paper. The application of our method to dynamical web documents also should be studied. Such studies should lead to a more general solution to web caching and replication problems.

References

1. Aggarwal, C., Wolf, J. L. and Yu, P. S. (1999) *Caching on the World Wide Web.* IEEE Transaction on Knowledge and Data Engineering, 35, 94-107.
2. Barford, P. and Crovella, M. (1998) *Generating representative web workloads for network and server performance evaluation.* Proc. of ACM SIGMETRICS'98, Madison, WI, June, pp. 151-160.
3. Bates, T., Gerich, E., Joncheray, L., Jouanigot, J. M., Karrenberg, D., Terpstra, M. and Yu, J. (1995) *Representation of IP routing policies in a routing registry.* Technical Report, Zvon-RFC 1786, May.
4. Bestavros, A. (1997) *WWW traffic reduction and load balancing through server-based caching.* IEEE Concurrency: Special Issue on Parallel and Distributed Technology, 15, 56-67.
5. Breslau, L., Cao, P., Fan, L., Phillips, G. and Shenker, S. (1999) *Web caching and zipf-like distributions: evidence and implications.* Proc. of IEEE INFOCOM'99, March, pp. 126-134.
6. Calvert, K. L., Doar, M. B. and Zegura, E. W. (1997) *Modelling internet topology.* IEEE Comm. Magazine, 35, 160-163.
7. Krishnamurthy, B. and Rexford, J. (2001) *Web Protocols And Practice.* Addison-Wesley, Boston.
8. Li, K. and Shen, H. (2004) *Dynamically Selecting Distribution Strategies for Web Documents According to Access Pattern*, Proc. of the Fifth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT 04), pp. 554-557.

9. Loukopoulos, T., Ahmad, I. and Papadias, D. (2002) *An overview of data replication on the Internet.* Proc. of the International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'02), Makati City, Metro Manila, Philippines, 22-24 May, pp, 31-37.
10. Pierre, G. and Makpangou, M. (1998) *CSaperlipopette!: a distributed web caching systems evaluation tool.* Proc. of 1998 Middleware Conference, The Lake District, England, 15-18 September, pp. 389-405.
11. Pierre, G. and Steen, M. (2002) *Dynamically selecting optimal distribution strategies for web documents.* IEEE Transactions on Computers, 51, 637-651.

Web-Based Authoring Tool for e-Salesman System

Magdalene P. Ting and Jerry Gao

San Jose State University, Computer Engineering, San Jose, CA 95192-0180
jerrygao@email.sjsu.edu

Abstract. Searching and finding items on the WWW is increasingly difficult for businesses and for consumers. Many navigation and keyword searches are inadequate for the modern consumer. What's plaguing e-commerce is the lack of intelligent assistance. The e-Salesman System (eSS) [11], based on a knowledge-driven intelligent model, aims to simulate the human element of traditional shopping for online sales. This paper presents a tool for authoring and managing an intelligent system that will change the current approach to online browsing, searching, and shopping. The contribution of this solution is to allow merchants to customize and change their e-retail shops to interact with online users based on their dynamic changed models to meet their business rules and marketing needs.

1 Introduction

E-commerce offers many advantages to both buyers and sellers alike. However, in reality, online sales are not what they were projected to be. While e-commerce has grown vastly with the great many advantages it offers both sellers and buyers, it contributes only to a very small percentage of total sales. Many are seen to use the Internet stores as more of a research tool than an actual store where they would buy products. The design of web-stores is one of the factors that influence browsers into becoming buyers. A number of credible surveys show that a large number of online shoppers feel many websites are difficult to navigate, and searching for the right products is not smooth or easy. For novice shoppers, being presented with many links, flashing text, attractive pictures, and numerous advertisements is often confusing and frustrating. A very important feature that many websites fail to show is a persona, the human element that exists in traditional shopping. We see the need of making websites more intelligent, personal, friendly, human, and less confusing while appearing more trustworthy to the customer, making the online shopping experience parallel to the traditional one.

To address the need for these missing factors in online shopping, an electronic salesman solution was developed in San Jose State University with the guidance of Dr. Jerry Gao. The e-Salesman System (eSS) is an intelligent system that allows creation of virtual-human sales representatives for web-stores. By providing intelligent sales, persona, and a human element to customer-website interaction, the eSS offers a cost effective solution to the current online shopping problem. Virtual sales representatives can be employed for sales, customer support, general information and guidance, and numerous other applications. The eSS is an adaptable

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 528 – 537, 2005.
© IFIP International Federation for Information Processing 2005

and an authorable system that can be customized to suit the needs and requirements of any web-store by providing the web-store own unique interaction model [1]. This paper talks about the architecture and design of the system by providing design models using UML. It presents an intelligent interaction and authoring solution that supports the interaction between the eSS, web-store administrators, and online customers.

2 Related Work

This section presents a brief summary of current existing work in the study and applications of web agents and artificial intelligence effects in e-commerce applications. In the recent years, a number of published papers addressed the needs of intelligent assistance to support e-commerce application systems. Caroline C. Hayes [6] discusses the escalating significance of agent-based approaches. Agents are changing the way people conduct businesses and manage information and investments. Nishida [7] and Takeuchi and Katagiri [8] discuss the personality or social characteristics of animated interface agents. People tend to expect personalized agents to behave intellectually, in the same way as humans. There is an increase of design approaches to intelligent authoring tools for tutoring and learning [9][10] but the focus on e-commerce is still scarce.

Current leading technology groups are increasing their efforts to produce "intelligent agents" to perform a number of web-based activities. Many companies see the need for employing sales and business intelligence on their business and retail portals. Currently, these forms of intelligence are seen in intelligent search agents, automated user driven product selection agents, animated personas, natural language search, intelligent advertising, intelligent recommendation, etc [2]. The more common forms of intelligence are seen in intelligent personalization, intelligent recommendation, and intelligent advertising [3].

The less common forms of intelligence are seen in the following areas:

1. *Intelligent interfaces*: Some companies, such as www.nativeminds.com, do offer intelligent interface solutions.
2. *Intelligent search*: www.askjeeves.com is a search interface that provides searching capabilities with natural language inputs.
3. *Intelligent sales assistance*: www.liveperson.com is one company that tries to bridge the gap between human assistance and online assistance but it still requires human representatives in the background.
4. *Intelligent purchase assistance*: A. Burns and G. Madely discussed how web-based agents might be used to aid buyers in fashion selection and the online purchasing process [4].
5. *Intelligent customer services*: Aberdeen Group Inc. discussed the demand for increased online customer service and how automated agents may help meet these demands [5].

We see the need to develop an intelligent integrated solution that uses all the commonly seen and less commonly seen intelligent applications. The main reason is

to make online commerce a more human, personal and friendly experience that appears to mimic the traditional shopping experience for shoppers.

3 The e-Salesman System

This section contains an overview of the e-SS, including its architecture, high-level models, technology, rules, and features [11][12]. There are many qualities and traits that form a good sales person. A good and intelligent sales person should possess an intelligent aura and a likeable persona. This impression creates a welcoming and sincere environment to customers. He or she should be able to communicate using a common language so as to be clearly understood by customers and should possess ample knowledge about products, knowledge about clients, and strategic knowledge. The sales person should know how to sell the right product, to communicate with buyers, to recommend products, and to negotiate with customers.

E-Salesman

The eSS is designed to be an intelligent sales platform that employs all of the intelligent application areas in e-commerce. Unlike existing work, our goal is to develop an intelligent customer representative with human-like conversational capabilities, reasoning, knowledge, and persona. The platform will be based on knowledge driven model with knowledge-based reasoning and architecture. It can be authored to provide information services, as well as sales and business knowledge driven services. The purpose is to make e-commerce as natural, as effortless, and as intelligent as traditional shopping experiences.

The eSS has a set of requirements that makes it useful in its areas of application and must have a few basic qualities to achieve the expected results. The eSS has a *persona*. One of the main features that influence how users perceive intelligence is the interface. The use of animated characters and human persona has proven to be successful. The different expressions of a persona are shown in .Fig.1. The eSS must have *natural language support*; the conversation that a customer has with a salesman should be as seamless and effortless as an online chat or a phone call with an actual human representative. The eSS must possess *product knowledge* and be able to use and query that knowledge on the customer's request. The eSS must also be equipped with *sales knowledge*. It must be aware of any special offers, promotions, and over-stocked inventory and pricing, and should be able to use this information to make a successful sale. The eSS servicing a customer in a session must remember the customer, his or her needs, his or her stated queries and all other customer information throughout the session. The eSS needs *persistence* to keep track of the customer's navigation, choices, needs and purchases all throughout the customer's visit.

Fig. 1. Persona with Different Expression

System Architecture

The overall architecture for this system is a three-tier web-based system that includes the clients, a web server and application server and the database server. Fig. 2 shows the system architecture.

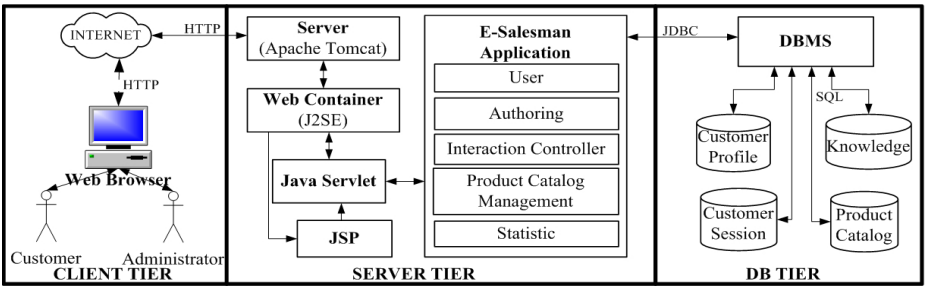


Fig. 2. Overall Architecture

Design for the Authoring Tool

Fig. 3 shows the authoring module design consists of a collection of components that allow the user to access the authoring services provided by the eSS. It consists of the knowledge data access component supporting the display servlet and the author controller component. The knowledge data controls access to the models, nodes, templates, tokens, and expressions that allows the access, author and management of the components. User can access and author model elements or node elements. Model elements include general properties and information of a model whereas node elements consist of general properties, seller properties, buyer properties, node condition, and URL. Buyer properties may also include one or multiple token values. The eSS's authoring tool is a subsystem that is adaptable and can be used with existing e-commerce websites. Our application was implemented with JAVA, JSP, Servlets, HTML, JavaScript, JDBC, and SQL. The web-server used was Jakarta Tomcat.

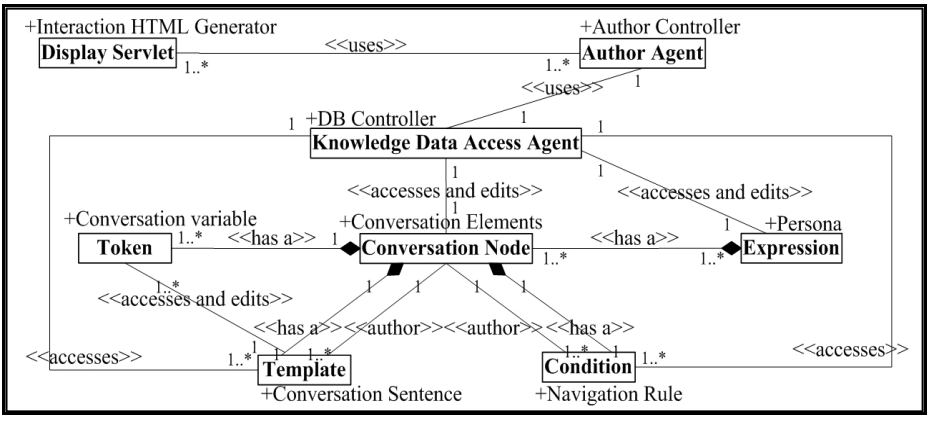


Fig. 3. Authoring Tool Design

4 The Authoring Solution

The eSS includes a comprehensive solution for the interaction between the system and the users and an intricate authoring solution for customization support for web-stores. In order to magnify the details of the authoring subsystem solution, we need to first proceed to give some details of the interaction of the eSS.

4.1 The Underlying Interaction Solution

The interaction structure of the eSS is an intelligent condition-based system. The interaction-processing piece includes the intelligent agent and the interaction controller. The interaction controller receives selected conversation template, with or without token values, from the customer through the user interface and dispatches it to the intelligent agent. The intelligent agent takes the information and dynamically accesses the knowledge repository for the condition that corresponds with the customer's selection. The new information will then be displayed by the user interface after receiving them from the interaction controller.

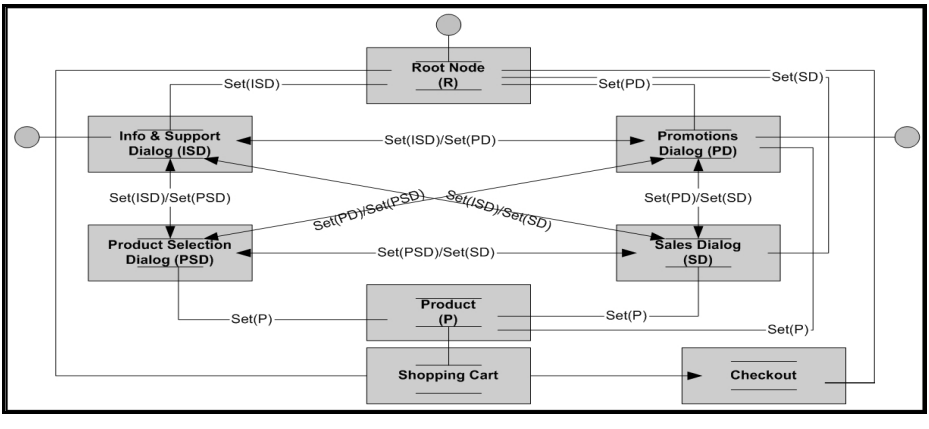


Fig. 4. Context-based Interaction Model

A conversational model is a graph M = (N, L), where N is a set of conversational nodes and L is a set of directed links. A single node, Ni = (CT, P, U, E, C), is a single conversation element that is an utterance from the salesman and customer. A link is a transfer of interaction from one node to another. A node consists of a set of conversational texts called templates, CT, a notice, P, a URL, U, a persona expression, E and a condition, C. A single template, CTi = (S, T, T), consists of a template text, S, a role, R and a set of tokens, T. A single token, Ti, is a variable in the template text that can take a value during the customer-salesman conversation. Examples of tokens are "make" or "model" or "manufacturer" or "price". A token, Ti = (m, f, y), consists of a token name m, a list filename f, which is a locator of the list of possible values, and a type y, which describes the nature of the token. The values that a token may take at any given point are dynamic data generated from a source. Every node has a condition, C

(1). The condition is a test case which returns a true or false value when executed. This value determines whether that particular conversation node is to be traversed during the conversation. The condition utilizes a number of variables that can be tested - the identity of the parent node, the values of tokens previously determined, and the conversation template picked by the customer in the parent node.

Fig. 4 presents a context-based interaction model for the eSS interaction. The model illustrates major conversation subjects for the interaction between the eSS and a shopper. These subjects can be classified into the following four major categories: sales-oriented dialog, support and information oriented dialog, promotion oriented dialog and product selection oriented dialog. The links between two subjects represents a transfer of conversation from one subject to another. From the figure, conversation may start from the root node, promotions dialog or support and information dialog. Conversation may transfer from one subject to another depending on the shopper or the eSS.

4.2 The eSS Authoring Solution

Fig. 5 shows the authoring processing component that includes the authoring agent and the data controller. The data controller controls receiving data from the user interfaces and handing it to the authoring agent. The authoring agent takes the information and appropriately processes it as creation or editing of node, token, or expression. The user interfaces are customized by the different elements that may be authored. The administrator, as the user, can author node, tokens and expressions through the respective interfaces. One of the main objectives of creating the authoring tool is to make the tool straightforward to author the elements; the different user interfaces make it simple for the user to author the different elements. The authoring processing component is designed with detailed rules that build and change the growth of the entire interaction repository. There are many different nodes that make up the entire sales conversation. The features of a node can be that of inquiry, informative, critical, or supplementary.

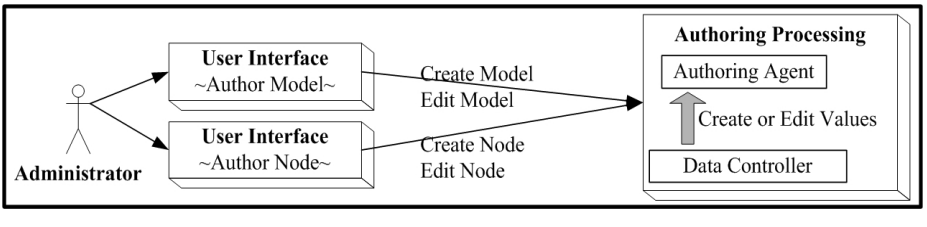


Fig. 5. Agent Architecture for Authoring

The Authoring Module

The authoring module is a detailed ensemble of helpful and user-friendly interfaces, complex rules for managing the collection of nodes and links that form a part of the entire sales-oriented conversation, and intricate algorithms for performing the intelligence of the authoring system. The authoring tool is designed to be rich in usability; methodical with accepting valid information and clever with the ability to

recognize errors while portions of the nodes and links and being authored. The algorithm supporting this feature is highly significant in this system because the information of conversations entered by a person can grow extremely vast and keeping track of accurate conversational input can very quickly be beyond human capabilities. Thus, this tool is complex yet highly supportive to the end user.

Authoring Application

Authoring the eSS system requires a combination of knowledge; sales, marketing, and some technicality of the eSS system. There are various ways to author the system for the model to represent an intended interaction between the eSS and a user. The next few figures will show one way for authoring a specific car dealership.

Fig. 6 show the model of how the system can be authored. The author starts with either creating a new model or loading an existing model and goes on to either editing model properties or authoring a node. The author may author a new node or edit an existing node. The node properties include the general properties, node condition, node URL, seller's template, buyer's template, and token values. While authoring the templates, the author can also create or edit tokens associated with a node. The author can end the authoring at several instances or continue in the iteration of authoring the node properties.

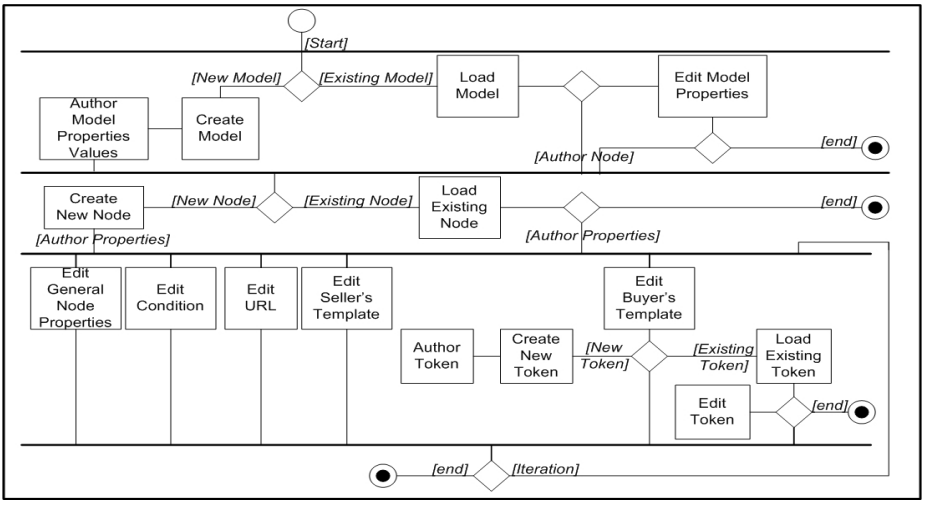
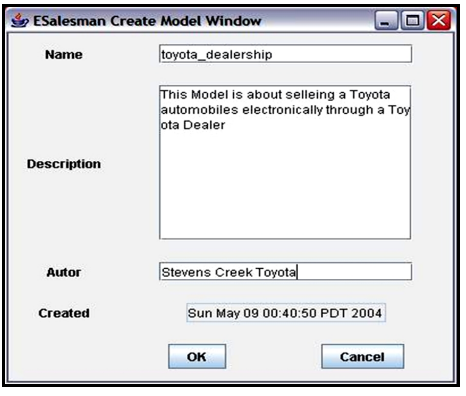


Fig. 6. Authoring Model

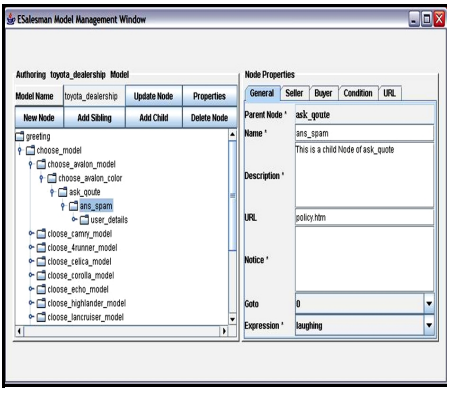
5 Application Example

Authoring GUI

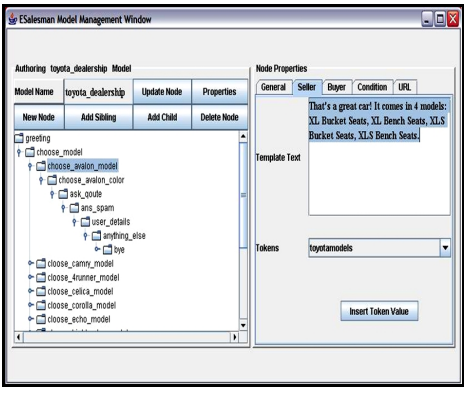
The authoring interface is one of the significant features of the authoring tool. It is the liaison between human and digital knowledge. The following figures show the subse-



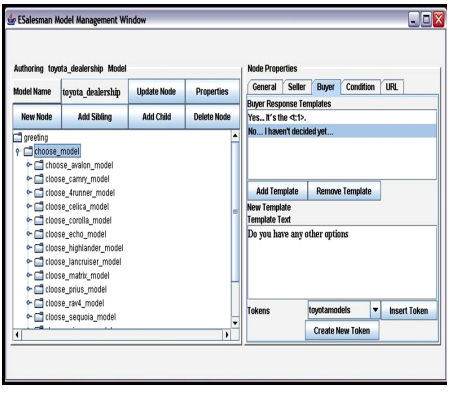
(a) Model Management Window



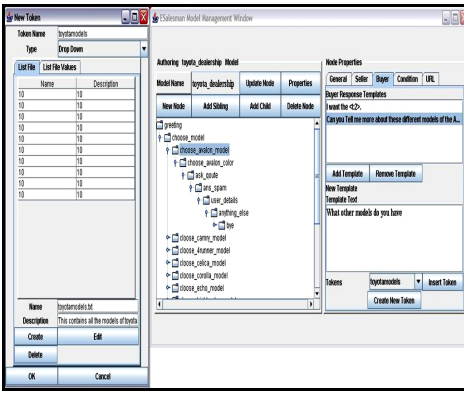
(b) Authoring General Node Properties



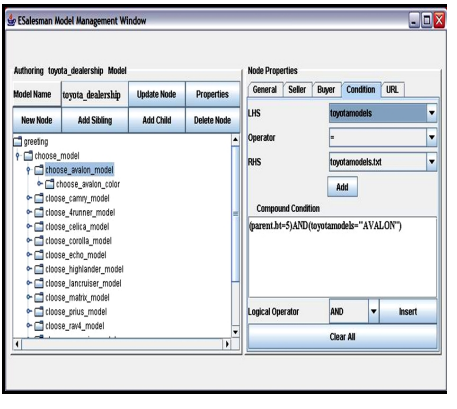
(c) Authoring Seller Template



(d) Authoring Buyer Template



(e) Authoring Token



(f) Authoring Node Condition

Fig. 7. Authoring GUI

quent GUI after loading an existing car dealership model. Fig 7(a) shows how to create a new model. In the Model Management GUI in Fig 7(b), the window is separated into two important sections; the Model Panel and the Node Properties Panel. These two panels allow the author to manage all the nodes in a tree view and also the information within a single node. Users can author or manage the conversational nodes and links in the Model panel. This panel gives a visual representation of the conversations nodes in a tree view. This visual feature is an organized way to help the author manage the entire conversation nodes.

The Node Properties panel allows the authoring of the content within a single node. This figure shows how a user can author general node properties. Users can author name, description, URL, notice, "goto" node and eSS's expressions for a node. Fig 7(c) and Fig 7(d) show how to author and manage the eSS and the buyer conversations by editing the seller's or the buyer's template. The author can also create or edit tokens within the buyer's template. Fig 7(e) illustrates the GUI when a new token is authored. The condition needed to traverse to a node is authored within the Node Properties panels. Fig 7(f) shows how tokens are linked to token files using Boolean expressions.

Interaction Example

As a test example for our eSS, we chose the online car sales industry. We specialize in selling new Toyota models. The website is like any other Toyota dealership website with numerous links, pictures, and advertisements But with an addition of a friendly salesperson face with a talk button that customers can talk with. On clicking the talk button, the customer is taken to the salesperson page which is a five-panel layout page shown in Fig 8. All interaction with the salesperson takes place on this page. The salesperson page consists of five panels: *the URL panel, the persona panel, the salesperson interaction panel, the customer interaction panel, and the notice panel.*

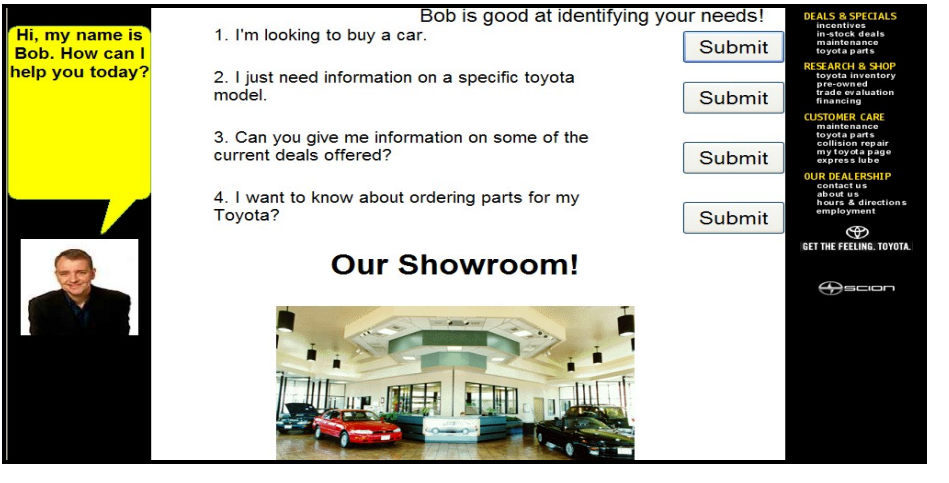


Fig. 8. Homepage

6 Conclusion and Future Work

In this paper, we propose an intelligent eSS solution to current e-commerce systems. We planned, designed, and implemented the eSS as an entity with an intelligent interaction engine. This eSS possesses persona, natural-language support, and intelligence for smart interaction with users. This system is a cost effective solution for companies wishing to enhance their customers' shopping experience. In addition, we extended our project with a complete design and implementation of a web-based authoring tool to provide data to the interaction component. This authorable component makes the eSS customizable to meet different industries' needs. Future work and research will include support for different clients, for marketing and sales intelligence, for intelligent advertisements, for intelligent negotiations and recommendations, and for voice and video enhancements.

References

1. Ryszard Kowalczyk, Mihaela Ulieru, and Rainer Unland, "Integrating Mobile and Intelligent Agents in Advanced e-Commerce: A Survey", Agent Technologies, Infrastructures, Tools, and Applications for E-Services, 2002.
2. Sanwar Ali, Monsurur Rahman, and Kustim Wibowo. Indiana University of Pennsylvania. "Artificial Intelligence and E-Commerce", *First Annual ABIT Conference*, Pittsburg, Pennsylvania, May 3-5, 2001.
3. Ronald R. Yager and Iona College, "Targeted E-Commerce Marketing Using Fuzzy Intelligent Agents", *2000 IEEE Intelligent Systems*.
4. A. Burns and G. Madey. "Development of a web-based Intelligent Agent for The Fashion Selection and Purchasing Process via Electronic Commerce", *Proceeding from Americas Conference on Information Systems*, 1998, pp. 140-141
5. Aberdeen Group Inc., "Interactive Customer Care: Enriching the Self-Service Experience with Automated Agents. An Executive White Paper", November 2000.
6. C. C. Hayes, "Agents in a nutshell- a very brief introduction", *Knowledge and Data Engineering, IEEE Transactions*, Vol. 11, No: 1, pp: 127 -132, 1999.
7. Toyoaki Nishida, "Social Intelligence Design for the Web", IEEE Computer, 2002.
8. Y. Takeuchi and Y. Katagiri, "ATR MI&C Res. Labs., Kyoto. Social character design for animated agents", *Proceedings of RO-MAN '99, 8th IEEE International Workshop,* Pisa, Italy. September 27-29, pp: 53-58, 1999.
9. José A. Macías and Pablo Castells, "An Authoring Tool for Building Adaptive Learning Guidance Systems on the Web", *Proceedings of the 6th International Computer Science Conference on Active Media Technology*, p.268-278, December 18-20, 2001
10. T. Murray, "Authoring Intelligent Tutoring Systems: An analysis of the state of the art", *International Journal of Artificial Intelligence in Education*, Vol. 10, pp. 89-129, 1999.
11. Magdalene Ting, Vishal Seth, and Jerry Gao, Ph.D., "The e-Salesman System", *CEC 2004 Proceedings - IEEE Conference on E-Commerce Technology*, p.353-357, July 6-9, 2004
12. William Korbe, Valerie Stanton, and Jerry Gao, "iES: An Intelligent Electronic Sales Platform", Proceedings *of International Conference on E-Commerce Research (ICECR2003)*. Dallas Texas, Oct. 23-26, 2003.

Agent-Community-Based P2P Semantic Web Information Retrieval System Architecture

Haibo Yu[1], Tsunenori Mine[2], and Makoto Amamiya[2]

Department of Intelligent Systems, {Graduate School[1], Faculty[2]} of Information Science and Electrical Engineering, Kyushu University
6-1 Kasuga-koen, Kasuga, Fukuoka 816-8580, Japan
{yu, mine, amamiya}@al.is.kyushu-u.ac.jp

Abstract. In this paper, we propose a conceptual architecture for a personal semantic Web information retrieval system. It incorporates semantic Web, Web service, P2P and multi-agent technologies to enable not only precise location of Web resources but also the automatic or semi-automatic integration of Web resources delivered through Web contents and Web services. In this architecture, the semantic issues concerning the whole lifecycle of information retrieval were considered consistently and the integration of Web contents and Web services is enabled seamlessly.

1 Introduction

1.1 Motivation

With ever-increasing information overload, Web information retrieval systems are facing new challenges for helping people not only locating relevant information precisely but also accessing and aggregating a variety of information from different resources automatically.

Currently, new technologies for enabling precise and automatic machine processing such as semantic Web and Web services are emerging and have attracted more and more attentions from academia and industry in recent years.

The semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation [4]. Currently, there are a lot of researches such as [10] [9] [24] trying to apply semantic Web technologies to Web information retrieval systems, but they all address only problems concerning certain phases or certain aspects of the total complex issues involved. There isn't any research addressing the semantic issues from the whole life cycle of information retrieval and architecture point of view. However, for the reasons we show below, we argue that it is important to clarify the requirements of a Web information retrieval system architecture to apply semantic web technology to it. First, we need to ensure the semantics are not lost sight of during the whole life cycle of information retrieval, including publishing, querying, accessing, processing, storing and reusing. Hence the interfaces involved in the whole life cycle of information retrieval tasks need to be re-considered. Second, efficient searching for high quality results is based on

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 538–549, 2005.
© IFIP International Federation for Information Processing 2005

pertinent matching between well-defined resources and user queries, where the matching reflects user preferences. Therefore the description of Web site capability and the way of submitting queries incorporating user preferences should consistently be considered from an architectural point of view.

Web service mechanisms provide a good solution for application interoperability between heterogeneous environments. It will provide a new way for accessing Web information and play a vital role in Web information retrieval activities in the future. However, the conventional "Web contents" resources target at human consumption but new "Web services" resources target at machine consumption. Thus they have been managed separately for publishing, discovering, accessing, and processing until now. On the other hand, in the semantic Web, contents are given well-defined meaning, and they are becoming such data that can be understood and processed by machine as well. As both Web contents and Web services will be consumed by machines, this introduces the possibility and necessity of managing them together in a personal Web information retrieval system.

In this paper, we propose a conceptual architecture for a personal semantic Web information retrieval system. It incorporates semantic Web, Web services, peer-to-peer and multi-agent technologies to enable not only precise location of Web resources but also the automatic or semi-automatic integration of hybrid semantic information delivered through Web contents and Web services.

1.2 Approach

The conceptual architecture of our semantic Web information retrieval system is constructed based on the following four main ideas.

First, "using peer-to-peer computing architecture with emphasis on efficient method for reducing communication load." As a centralized system has the bottle neck of accessing and its maintenance cost is expensive, scalable and decentralized P2P systems are receiving more and more attention especially in the research and product development for the open and dynamic Web environment. However, due to the decentralization, the performance comes to be a significant concern when a large number of messages are propagated in the network and large amounts of information are being transferred among many peers [14]. Hence, an efficient mechanism for reducing communication load with least loosing of precision and recall is very important in a P2P information retrieval system. We propose our Agent-Community-based Peer-to-Peer information retrieval method called ACP2P method [15]. On the other hand, as we noticed that generally users retrieve and re-use certain amount of information repeatedly for a high percentage, it is essential to store frequently used information locally for the user with an efficient retrieval mechanism. We enable users refining and storing retrieved Web information in their local environment and manage them with semantic Web technology for the later re-use. As the user interested information is limited resource and the storing and retrieving mechanism can be adjusted to the user-specific way, the access time to the most frequently used information will be decreased significantly than searching in the vast open Web. As the

possibility of accessing external resources is decreased, the searching time and network transfer time are all saved.

Second, "all participants contribute to the semantic description consistently." The Web information retrieval system concerns three main kinds of participants: the "consumer" which searches for Web resources, the "provider" which holds certain resources, and the "mediator" which enables the communication between the consumer and the provider. In order to guarantee semantic interoperability during the whole life cycle of information retrieval, all participants need to consistently contribute to the semantic description. The provider needs to precisely describe their capabilities and the users need to pertinently describe their requirements as well. The mediator needs to correctly interpret the semantic dimension and to ensure that semantics are not lost sight of during the processing.

Third, "integrating Web contents with Web services." As we mentioned earlier, Web services will provide a new way for retrieving Web information. In fact, Web users do not care about how the system discovers, accesses and retrieves information from what kind of resources, they only care about the final results which can be used by them conveniently. Hence, the particular characteristics and the concrete realization details of both Web services and Web contents need to be hidden from users as much as possible. Therefore an integrated or unified management of Web contents and Web services needs to be carried out through different levels including the description of capabilities and requirements, querying, discovering, selection and aggregation.

Fourth, "providing a gateway to all the information that the user is interested in." A user generally concern two types of information: local and remote information. The local information such as documents, emails, contacts and schedule are stored at user's desktop and managed by various of applications, and the remote information such as Web information which are published by its Web site manager and can be searched and accessed through Web applications. Since the user needs to access and process all these local and remote information, a gateway providing an unified interface to all relevant information is necessary. We propose a personalized "Myportal [26]" to satisfy all the information requirements of a user.

The rest of the paper is organized as follows: Section 2 outlines our conceptual architecture, the components and their communication mechanism of a personal semantic Web information retrieval system. The process flow of an information retrieval system will be described in section 3. In Section 4, we will discuss related work and the concluding remarks will be summarized in section 5.

2 A Conceptual Architecture

Our conceptual architecture for a personal semantic Web information retrieval system is illustrated in figure 1.

The architecture consists of three main components: "consumer" which searches for Web resources, "provider" which holds certain resources, and mediator which enables the communication between the consumer and the provider.

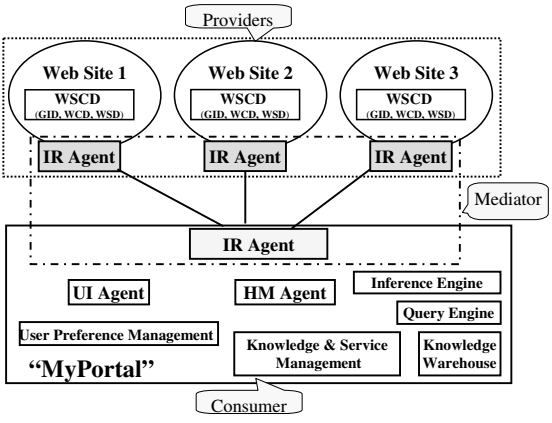


Fig. 1. A Conceptual Architecture

In our architecture, each provider describes its capabilities in what we call a WSCD (Web site capability description), each consumer is constructed as a "Myportal" providing a gateway to the information relevant to the user. The mediator is conformed by agents assigned to the consumer and providers using Agent-Community-based P2P information retrieval method to fullfill the searching and accessing tasks. We will describe each component of our architecture in a little more detail in following sub-sections.

2.1 Web Site Capability Description (WSCD)

Resource location is based on matching between user requirements and Web site capabilities, so a capability description of Web sites is necessary. We describe the layered capabilities of a Web site as shown in figure 2.

First, we semantically describe the general capabilities of the Web site, and we call this a "general information description (GID)." We argue that some explicit general ideas about a Web site are strongly required in order to precisely locate

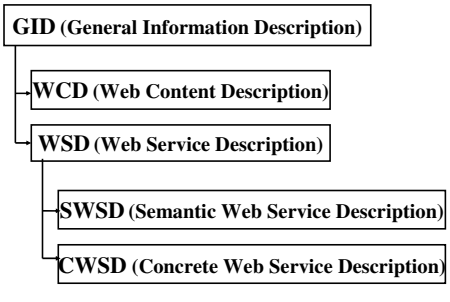


Fig. 2. Web Site Capability Description

Web resources based on user preferences. Therefore a brief general information description of the Web site is defined at the top level. The GID gives an explicit overview of the Web site capabilities, and can be used as the initial filter for judging congruence with user preferences.

Second, we give the Web content capability description (WCD) and there is a link from GID to WCD for using semantic Web contents. The WCD is the metadata of Web contents and is composed of knowledge bases of all domains involved. We use OWL [13] to describe domain ontologies and the metadata will be described in RDF [12].

Third, we give Web service capability description (WSD) and there is a link from GID to WSD for facilitating the further matching and use of Web services. In order to semantically describe the capabilities and support the concrete realization of services, we express the service capability description in two layers: "semantic Web service description (SWSD)" and "concrete Web service description (CWSD)." This hierarchical capability-describing mechanism enables semantic and non-semantic Web service capability-describing and matchmaking for different levels. we use WSDL [7] for the concrete Web service description and OWL-S [6] to express the semantic Web service description.

For the details of our Web site capability description mechanism, one can refer to document [27].

2.2 "Myportal"

"Myportal" is a "one stop" that links the user to all the information s/he needs. It resites on the user's own desktop which is also a Web server itself and is designed to satisfy user's personal information requirements and to be mastered freely by the user her/himself. It provides both semantic browser and semantic search engine functionalities, and these functions manage not only local user information but also the other Web sites as conventional browser. The information

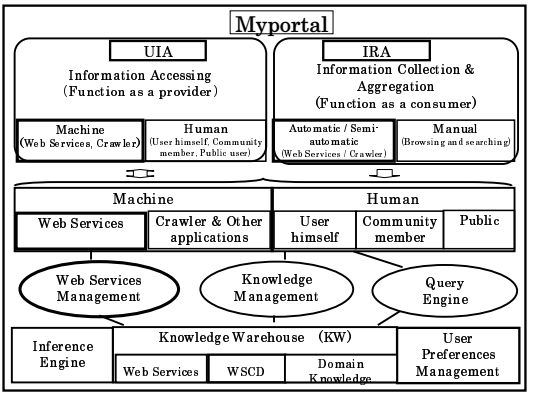


Fig. 3. Structure of "Myportal"

can be shared by others with proper authority. The structure of "Myportal" is shown in figure 3.

"Myportal" is composed of three types of main functional components: core component, consumer component and provider component.

The core component provides basic support for semantic technologies and information management. It consists of "Knowledge Warehouse (KW)," "Knowledge Management," "Query Engine (QE)" and "Inference Engine (IE)." As a consumer, it will bring together a variety of necessary information from different resources automatically or semi-automatically for the user. It is assigned agents to fulfill the information retrieval tasks through the communication with provider agents. As a provider, the contents and services of "Myportal" can be consumed by humans as well as machines. The human can be the user or other permitted persons, and the machine can be local or remote. A unified interface for browsing, searching and facilitating Web contents and services will be provided. We described "Myportal" in a little more detail in document [26].

2.3 Mediator

The communication between consumer and providers is based on an Agent-Community-based Peer-to-Peer information retrieval method called ACP2P method, which uses agent communities to manage and look up information related to a user query.

In order to retrieve information relevant to a user query, an agent uses two histories: a query/retrieved document history (Q/RDH for short) and a query/sender agent history (Q/SAH for short). Making use of the Q/SAH is expected to have a collaborative filtering effect, which gradually creates virtual agent communities, where agents with the same interests stay together. We have demonstrated through several experiments that the method reduced communication loads much more than other methods which do not employ Q/SAH to look up a target agent, and was useful for creating a "give and take" effect, i.e., as an agent receives more queries, it acquires more links to new knowledge[16].

The ACP2P method employs three types of agents: user interface (UI) agent, information retrieval (IR) agent and history management (HM) agent. A set of three agents (UI agent, IR agent, HM agent) is assigned to each user.

The UI agent receives requirements from the user, factors in missing or inherent information based on user preferences, breaks and transforms the requirements into formal queries and sends them to the IR agent.

When receiving a query from a UI agent, an IR agent asks an HM agent to look up target agents with its history or asks a portal agent to do it using a query multicasting technique. When receiving a query from other IR agents, an IR agent looks up the information relevant to the query from its original content and retrieved content files, sends an answer to the query-sender IR agent, and also sends a pair of the query and the address of the query-sender IR agent to an HM agent so that it can update Q/SAH.

The returned answer is either a pair of a 'Yes' message and retrieved information or a 'No' message indicating that there is no relevant information. When

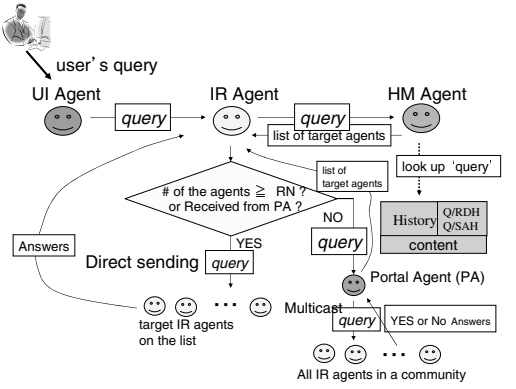


Fig. 4. Actions for Sending a Query

receiving answers with a 'Yes' message from other IR agents, the IR agent sends them to a UI agent, and sends them with a pair of a query and the addresses of answer sender IR agents to an HM agent.

The ACP2P method is implemented with Multi-Agent Kodama (Kyushu university Open & Distributed Autonomous Multi-Agent) [28]. Kodama comprises hierarchical structured agent communities based on a portal-agent model. A portal agent is the representative of all member agents in a community and allows the community to be treated as one normal agent outside the community. A portal agent has its role limited in a community, and the portal agent itself may be managed by another higher-level portal agent. A portal agent manages all member agents in its community and can multicast a message to them. Any member agent in a community can ask the portal agent to multicast its message. Fig.5 shows the agent community structure which the ACP2P method is based on.

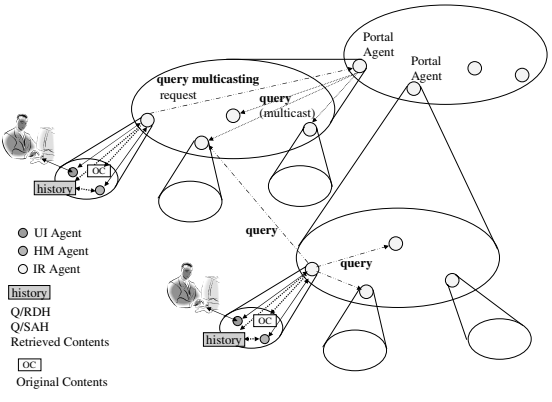


Fig. 5. Agents and their Community Structure

The query language and protocol communicated between IR agents need to be defined. Since the semantic Web information is commonly based on RDF which is a recommendation of W3C, a standard interface for querying and accessing RDF data is ideal for the interoperability between heterogeneous semantic Web information environments. The W3C RDF Data Accessing Working Group (DAWG) has published their working drafts of RDF Query Language SPARQL [19] and SPARQL protocol that are expected to be standards in this field. Although our architecture is designed for any reasonable communication interfaces, we are currently planning to use SPARQL RDF query language and SPARQL protocol as our semantic communication interfaces between providers and consumers.

3 Process Flow of Information Retrieval

The total process flow of the P2P Web information retrieval system can be illustrated as shown in figure 6.

At the beginning, the consumer (the user) will edit his profile and preferences, and providers will describe their capabilities using WSCD which we described in section 2.1. When the user wants to search for the information, a search interface need to be provided. Although various kinds of user interface, such as natural language, template-style or formal equation, can be considered, taking user convenience and reality in account, we provide a template-style search interface, enabling users to input or select their preferences as well as query items from

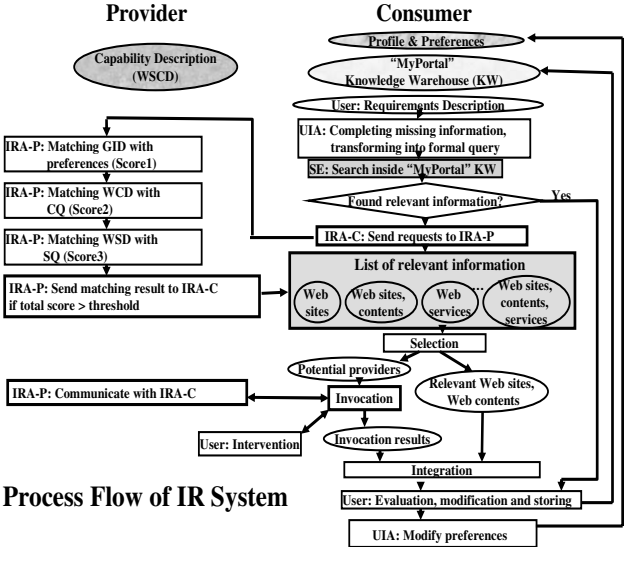


Fig. 6. Process Flow

recommendation lists. The missing or inherent information will be inferred based on the user profile and preferences, and the requirements will be broken down and transformed into formal queries. The formal query is composed of three types of element fields: user preferences (UP), content query (CQ) and Web service query (SQ). The search will be carried out first inside the MyPortal knowledge warehouse, and only when we cannot find satisfactory information from MyPortal, will we extend the search to the other providers and the request will be sent to the candidate information retrieval agents on provider side (IRA-P for short) through the information retrieval agent on consumer side (IRA-C for short). The information discovery on the provider side is based on matching between user requirements and provider capabilities. We do matching at three levels. First, we do matching of Web site general description (GID) against user preferences to see whether they match at the overview level or not. Second, we do matching of Web contents, and finally do the matching of Web services. A matching score will be given from the matching of each level and they will be used for the final judgment of relevance of Web contents and Web services. IRA-Ps will send back their matching scores to the IRA-C, and the IRA-C will judge and select the most relevant Web services and Web contents based on a total consideration of those matching scores. After selecting the most relevant Web services, the IRA-C will invoke those services. If the input information is not sufficient to trigger invocation, the IRA-C will request the user to provide necessary information through the UIA. The results from different Web service invocations as well as the Web contents results will be aggregated by the IRA-C into a refined, final result based on user preferences and be sent to the user through the UIA. This result can be evaluated, modified and stored in the user's MyPortal knowledge warehouse for future reuse. The integration of different Web service invocation results and Web contents is based on their common RDF data model.

4 Related Work

In this section, we discuss some related work that is directly or indirectly of interest to our research work.

Francisco et al. [18] presented an architecture for an infrastructure to provide interoperability using trusted portals and implemented such an infrastructure based on Thematic Portals. The searching portals use semantic access points based on metadata for more precise searching of the resources associated with the potential sources of information. The proposed architecture supports specific and cross domain searching, but only provides semantic representation for the capabilities of Web contents not for their services as far as we understand. Our semantic Web site capability description and pertinent user requirements and preferences description provide interoperability for both Web contents and Web services.

RSS [21] and Atom [17] are lightweight multipurpose extensible metadata descriptions and syndication formats. FOAF vocabulary [5] provides a collection of basic terms that can be used in machine-readable Web homepages for people, groups, companies and so on. RSS, Atom and FOAF vocabulary all focus on cer-

tain kinds of Web contents description such as news, Web blog or people, they do not include Web services as we proposed. Our Web site capability description describes not only Web contents but also Web services, so the resources of the portal can not only be located but also used as a computational part of the information retrieval system. RSS, Atom and FOAF can be used for the Web contents capability description which is a part of our Web site capability description.

There are Web portals based on Semantic Web technology, such as KA2 [1] and SEAL [24], but they target uniform access by large numbers of people for human navigation and searching. SEAL provided an interface for a software agent but only for a crawler. None of them supports Web services for information aggregation and publishing at present, as far as we know. Our "Myportal" is a personalized gateway to all user-relevant information and it not only aggregates Web information but also shares its information through Web services.

Haystack [10] and Gnowsis [22] are semantic Web enhanced desktop environment for personal information management. The main purpose of them is to semantically manage user's local information enabling an individual to flexibly manipulate his/her information with a personalized way. They are not constructed from the Web portal point of view and doesn't emphasize the support of machine interoperability between users enabling Web service functionalities. We refer to their ideas of personalization on information management and the integration of existing desktop applications, construct our semantic personal information system as a fully personalized Web portal to provide a gateway to access to not only the local personal information but also the Web information. The "Myportal" acts as both a consumer and a provider to form a basic unit of a P2P information retrieval system. The Web services will be used not only for information retrieval but also for information delivery.

There is lots of work related to the ACP2P method. Structured P2P networks associate each data item with a key and distribute keys among directory services using a Distributed Hash Table (DHT)[23, 20, 25]. Hierarchical P2P networks use the top-layer of directory services to serve regions of the bottom-layer of leave nodes, and directory services work collectively to cover the whole network[8, 2, 3, 11]. The common characterlistics of both approaches are the construction of an overlay network to organize the nodes that provide directory services for efficient query routing. The ACP2P also employes a hierarchical P2P computing architecture based on the Multi-Agent Kodama [28] framework. Unlike other work, the ACP2P makes use of only local information : retrieved documents and two histories, Q/RDH and Q/SAH for query routing. Especially, the Q/SAH is an important clue to search for the relevant information. Further the characteritics using only local information makes the ACP2P responsive to the dynamic environment where the global information is not available.

5 Conclusion

In this paper, we addressed the main aspects of a semantic Web information retrieval system architecture trying to answer the requirements of next-generation

semantic Web users. In the architecture, the semantic issues and the integration of Web contents and Web services are considered for the whole lifecycle of information retrieval. Our "Myportal" aims at constructing a fully personalized user's local Web portal, which is adapted to user preferences and satisfies all the requirements of a user's local and Web information usage. We use the ACP2P method for the communication between consumer and providers, which uses agent communities to manage and look up information related to a user's query in order to reduce communication loads in a P2P computing architecture.

In the future, we will realize a prototype of an agent community based P2P personal semantic Web information retrieval system, and evaluate the effectiveness of our proposed architecture based on it. Further experiments for the ACP2P method on semantic Web data retrieval need to be done to see its effectiveness.

References

1. KA2 Portal. http://ka2portal.aifb.uni-karlsruhe.de/.
2. Kazaa v3.0. http://www.kazaa.com/.
3. M. Bawa, G. S. Manku, and P. Raghavan. Sets: search enhanced by topic segmentation. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 306 – 313, 2003.
4. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May, 2001.
5. D. Brickley and L. Miller. FOAF Vocabulary Specification. Sept., 2004.
6. David Martin et al. OWL-S 1.1 Release, November, 2004. http://www.daml.org/services/owl-s/1.1/.
7. Erik Christensen et al. Web Services Description Language (WSDL) 1.1, March 15, 2001. http://www.w3.org/TR/wsdl.
8. Gnutella. http://gnutella.wego.com/, 2000.
9. R. Guha, R. McCool, and E. Miller. Semantic Search. In *Proceedings of WWW2003*, pages 700–709, 2003.
10. D. Huynh, D. Karger, and D. Quan. Haystack: A Platform for Creating, Organizing and Visualizing Information Using RDF. In *Proceedings of the International Workshop on the Semantic Web (at WWW2002)*, 2002.
11. J. Lu and J. Callan. Content-based retrieval in hybrid peer-to-peer networks. In *Proceedings of the twelfth international conference on Information and knowledge management*, pages 199–206, 2003.
12. F. Manola and E. Miller. RDF Primer, February 10, 2004. http://www.w3.org/TR/rdf-primer/.
13. D. L. McGuinness and F. van Harmelen. OWL Web Ontology Language Overview, February 10, 2004. http://www.w3.org/TR/2004/REC-owl-features-20040210/.
14. D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu. Peer-to-Peer Computing. Technical report, HP, 2002. http://www.cs.wpi.edu/ claypool/courses/4513-B03/papers/p2p/p2p-tutorial.pdf.
15. T. Mine, D. Matsuno, A. Kogo, and M. Amamiya. Acp2p : Agent community based peer-to-peer information retrieval. In *Proc. of Third Int. Workshop on Agents and Peer-to-Peer Computing (AP2PC 2004)*, pages 50–61, 7 2004.

16. T. Mine, D. Matsuno, A. Kogo, and M. Amamiya. Design and implementation of agent community based peer-to-peer information retrieval method. In *Proc. of Eighth Int. Workshop CIA-2004 on Cooperative Information Agents (CIA 2004), LNAI 3191*, pages 31–46, 9 2004.
17. M. Nottingham. The Atom Syndication Format 0.3 (pre-draft), December, 2003. http://www.mnot.net/drafts/draft-nottingham-atom-format-02.html.
18. F. Pinto, C. Baptista, and N. Ryan. Using Semantic Searching for Web Portal Interoperability. In *International Workshop on Information Integration on the Web - Technologies and Applications, April 9-11, Rio de Janeiro - Brazil*, April 2001.
19. E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF, April 19, 2005. http://www.w3.org/TR/rdf-sparql-query/.
20. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *SIGCOMM*, pages 161–172, 2001.
21. RSS-DEV Working Group. RDF Site Summary (RSS)1.0, 2000-12-06. http://web.resource.org/rss/1.0/.
22. L. Sauermann. The Gnowsys Semantic Desktop for Information Integration. In *IOA Workshop of the VM2005 Conference*, 2005.
23. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*, pages 149–160, 2001.
24. N. Stojanovie, A. Maedche, S. Staab, R. Studer, and Y. Sure. SEAL – a framework for developing SEmantic PortALs. In *Proceedings of the International Conference on Knowledge Capture*, pages 155–162, 2001.
25. C. Tang, Z. Xu, and S. Dwarkadas. Peer-to-peer information retrieval using self-organizing semantic overlay networks. In *SIGCOMM*, 2003.
26. H. Yu, T. Mine, and M. Amamiya. Towards a Semantic MyPortal. In *The 3rd International Semantic Web Conference (ISWC 2004) Poster Abstracts*, pages 95–96, 2004.
27. H. Yu, T. Mine, and M. Amamiya. Towards Automatic Discovery of Web Portals -Semantic Description of Web Portal Capabilities-. In *Semantic Web Services and Web Process Composition: First International Workshop, SWSWPC 2004, LNCS 3387/2005*, pages 124–136, 2005.
28. G. Zhong, S. Amamiya, K. Takahashi, T. Mine, and M. Amamiya. The Design and Implementation of KODAMA System. *IEICE Transactions on Information and Systems*, E85-D(4):637–646, April, 2002.

A Scalable and Reliable Multiple Home Regions Based Location Service in Mobile Ad Hoc Networks

Guojun Wang[1], Yingjun Lin[1], and Minyi Guo[2]

[1] School of Information Science and Engineering, Central South University, Changsha, P.R. China, 410083
[2] School of Computer Science and Engineering, University of Aizu, Aizu-Wakamatsu City, Fukushima 965-8580, Japan
csgjwang@mail.csu.edu.cn, linhandsome@126.com, minyi@u-aizu.ac.jp

Abstract. Compared with topology-based routing, location-based routing scales much better in large-scale mobile ad hoc networks. Location-based routing protocols assume that a location service is available to provide location information of each node in the network. Many location service protocols have been proposed in the literature. However, either they do not scale well in large-scale network environment, or they are not reliable if the network is highly dynamic. We propose a multiple home regions based location service protocol in large-scale mobile ad hoc networks. Theoretical analysis shows that the proposed protocol outperforms existing protocols in terms of both scalability and reliability.

Keywords: Mobile ad hoc networks, location service, location-based routing, scalability, reliability

1 Introduction

In recent years the widespread usage of wireless communication and handheld devices has stimulated research on self-organizing networks. Mobile Ad hoc NETworks (MANETs) are self-organizing, rapidly deployable and dynamically reconfigurable networks, which are formed by mobile nodes with no pre-existing and fixed infrastructure. Usually, these mobile nodes function as both hosts and routers at the same time. Two mobile nodes communicate directly if they are within the radio transmission range of each other; otherwise, they reach each other via a multi-hop route. Some typical applications in MANETs include communication in battlefield and disaster relief scenarios, video conferencing and multi-party gaming in conference room or classroom settings.

To route packets is one of the fundamental tasks in MANETs, but it is very challenging because of the highly dynamic topology of the network triggered by node mobility. There are two different approaches to route packets in such a network environment, namely, topology-based routing and location-based routing [1]. Topology-based routing protocols use the information about the communication links that are available in the network to perform packet forwarding. Due to node mobility, topology-based routing protocols can not scale well in large-scale MANETs. In

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 550 – 559, 2005.
© IFIP International Federation for Information Processing 2005

location-based routing, however, each node determines its own location information through the use of Global Positioning System (GPS) or some other type of positioning service. A location service, also known as mobility tracking or mobility management, is used by the sender of a packet to determine the location of the destination and to encapsulate it in the header of the packet. The routing decision at each forwarding node is then based on the locations of both the forwarding node's neighbors and the destination node. In this way, the location-based routing does not need to maintain routing tables as topology-based routing does. Therefore, location-based routing can scale quite well in large-scale MANETs [2] [3]. One of the main challenges of a location-based routing protocol is how to get the location information of a packet's destination when needed. Most of these protocols have a location service responsible for accomplishing this task. When a node does not know the location of its correspondent node, it requests the location information from a location service. Generally speaking, each node determines its own location information through the use of GPS or other techniques for finding relative coordinates based on signal strengths [4]. Since it is not necessary to maintain explicit routes in such protocols, location-based routing can scale well in large-scale MANETs even if the network is highly dynamic. This is a major advantage in MANETs where the topological change may occur frequently due to node mobility.

The rest of this paper is organized as follows. Section 2 introduces some related works. Section 3 presents our location service, which is used to update, maintain and query the location information of mobile nodes. Section 4 analyzes the scalability and reliability of the proposed protocol. Finally, we conclude this paper in Section 5.

2 Related Works

Location service is essential for designing location-based routing protocols in large-scale MANETs. Many location service protocols for MANETs have been proposed in the literature as follows.

Distance Routing Effect Algorithm for Mobility (DREAM) [5] is highly reliable and it provides localized information. The search of location of a destination requires only a simple local lookup. However, as the location information is periodically flooded into the whole network, the communication complexity is very large. DREAM thus has poor scalability and is inappropriate for large-scale MANETs.

Quorum-based approach is proposed in [6]. The key merit is the distribution of responsibility among quorums. But the quorum system has a major drawback in that it depends on a non-location-based routing protocol to maintain the integrity of the databases of the entire quorum, the implementation complexity of which is very high in MANETs. In particular, this drawback may greatly reduce the scalability.

To solve the scalability problem, the home-region based location service is proposed. In such a scheme, nodes within some geographical areas maintain the location information of other nodes which have made that area as their home region.

Similar to the Mobile-IP scheme [7], home-agent based location service (HALS) is proposed in [8]. This scheme eliminates the quorum system's major drawback. However, such a scheme is also not perfect due to the following reasons. (1) Since nodes can be hashed to any arbitrarily distant region, it may result in increased communication

complexity. (2) Since nodes only store location information in the nodes that are in the home agent region, if all the nodes in a home agent region which store a particular node's location information failed or left the home agent region, then the other nodes can not obtain this node's location information. That is, such a scheme is not reliable when a home agent region becomes empty due to the fact that all the nodes in the region become faulty or move out of the region simultaneously.

To solve the problems mentioned above, another home-agent based location service called the SLURP protocol has been presented in [9]. Although SLURP handles the problem of empty home region, there are some disadvantages as follows. (1) If the distance between a source node and a destination node is very close, but the source is far away from the destination's home region, the communication complexity will be very high. (2) If a node moves out of a region and it happens to be the last node in the region, then it needs to inform all the eight neighboring regions, and the overhead of which is very high. (3) Even worse, if the last node that moves out of the region happens to become faulty, it will lose the location information which is stored in this region, thus resulting in reduced reliability. (4) There is an extreme case in MANETs that also results in reduced reliability. That is, if one region becomes empty, then the eight regions surrounding the empty region will also become empty. Such an extreme case will not occur too frequently, but it does occur sometimes. For example, a bombing in the battlefield may damage the region where it occurs, and it may also affect all its neighboring regions of the damaged region.

Although the two protocols mentioned above scale well, their reliability may not meet requirements of some applications due to the existence of empty home regions. To improve the reliability of a location service, the GRID Location Service (GLS) is proposed in [10]. GLS is a promising distributed location service. However, the behavior of GLS in a dynamic environment and in the presence of node failures is difficult to control. Moreover, its implementation complexity is very high.

The SLALoM protocol, presented in [11], is similar to GLS. It improves both the query efficiency and the reliability in the sense that the use of both the near and far home regions reduces update traffic. However, the update traffic is still too high due to the fact that so many home regions are used for each node.

In order to reduce the update traffic, especially for those nodes which are not being queried, the ADLS protocol [12] adopts an adaptive demand-driven approach. Although ADLS reduces the update traffic, it affects querying efficiency. Even worse, when the primary home region becomes empty, the location information stored in this region will be lost, resulting in the reliability problem like the SLURP protocol.

In order to maintain the location information, the GLS, SLALoM and ADLS protocols have to set a lot of home regions for each mobile node in the whole network. Generally speaking, they improve the reliability compared with the HALS and SLURP protocols, but their scalability is worse than the HALS and SLURP protocols because so many home regions are used. Moreover, these protocols are too complex to implement in the highly dynamic MANETs. In order to provide a scalable and reliable location service, we propose a new location service protocol with multiple home regions, which can be considered as a tradeoff between the home-agent based protocols such as HALS and SLURP, and the GRID-based protocols such as GLS, SLALoM, and ADLS.

3 Overview of the Proposed Protocol

We propose a scalable and reliable Multiple Home Regions based Location Service (MHRLS) protocol for location-based routing in large-scale MANETs. In MHRLS, multiple home regions are assigned to each node by mapping its node ID. And all the nodes located in these regions are responsible for maintaining the approximate location information of the node to be mapped. To send messages from a source node to a destination node, the source node first queries the current location information of the destination node by MHRLS. After getting the location information, the source node sends messages to the destination by some location-based routing protocol such as the MFR protocol [13]. In this section, we overview the proposed protocol from the following four aspects, namely, dividing the large network into small regions, assigning home regions to each mobile node, and the update, maintenance and querying of the location information when needed.

3.1 Dividing the Large Network into Small Regions

We assume that each node in MANETs is equipped with GPS to get its accurate location information. Though it brings in extra expenses, it gains more by using location information. Each node has a unique node ID. A large rectangular area is divided into small rectangular regions. Each small region is assigned a unique region ID. An example network which is divided into 6*6 small regions is shown in Figure 1. Each node in the network is aware of the information about how the network has been divided and which small region itself belongs to.

3.2 Assigning Home Regions to Each Mobile Node

Before a source node *S* sends messages to a destination node *D* using a location-based routing protocol, it has to get node *D*'s current location while the only information it knows about node *D* is its ID. To solve this problem, node *S* can either probe the information by flooding or querying some other nodes who know the location where node *D* is. Obviously, the querying scheme might be more efficient than the flooding scheme in most cases. For the querying scheme, node *D* needs to first designate some nodes, or called its location servers, and then update the location servers with its location information. In MHRLS, a hash function is used to map each node in the network to all the nodes which are located in multiple home regions as its location servers.

More specifically, the MHRLS protocol establishes k functions in advance, each of which can map the same node ID into a different region ID (k is set as a system parameter) as follows: $f_i(Node\ ID) \rightarrow Region\ ID_i$

All nodes within the node's k home regions should maintain the node's current location information dynamically. Take node *D* in Figure 1 as an example, its k home regions are region 8, region 17 and region 26 (here k=3).

In order to make the k home regions to be evenly distributed in the whole network, the function f needs to satisfy the following two properties:

1. Function *f* can evenly map the node ID into every region in the whole network, i.e., the probability of being a home region is the same for every region in the network.
2. Function *f* can be used in MANETs with various shapes and different coverage sizes, i.e., the function still works even when the network shape and size change.

3.3 Location Information Update

After a node moving out of the current region, it first gets its home regions by k functions ($f_i(ID), 0 \le i \le k-1$), then the location update message (including node ID and current region ID) is sent to the centers of these k home regions separately. We assume to use some routing strategy based on geographic location information such as MFR protocol [13] to forward such a message. If the node which received the update message is not in the destination home region, it will forward the message; otherwise, it will broadcast the message to the rest of nodes within the destination home region. Finally, each node has a copy of its current location information stored in all the nodes of the k home regions. Take node D in Figure 1 as an example, after moving from region 23 to region 29, it will send an update message including its node ID and region 29's region ID to region 8, region 17 and region 26, respectively.

3.4 Maintenance of the Location Information

When a node moves into a new region, the node sends a message to its neighbors requesting location information registered in this region. Any neighbor which has such location information generates a reply message to the node, and then the node uses the reply message to maintain its new location information for the new region.

3.5 Querying the Location Information

A source node computes home region IDs of a destination node and sends a query message for the destination node's location. The proposed MHRLS protocol provides two kinds of queries for the location information:

1. The source node queries all home regions of the destination node, and we call such a scheme Query-All. More specifically, the source node sends one copy of the query message to the center of each home region. The first node receiving the message in the home region sends a reply message (including ID of the region where the destination node currently locates). To prevent the source node from receiving k reply messages, the source node will simply discard the following reply messages associated with the same query message after receiving the first one. This scheme is easy to realize and has high reliability, but the communication overhead is relatively high.
2. The source node queries the nearest one of the k home regions of the destination node, which is called the Query-Nearest scheme, and sets a timeout at the same time. The first node receiving the query message in the destination region sends a reply message. If the source node does not receive any reply message after the timeout period, it will send a query message to the nearest one out of the rest k-1 home regions of the destination node. This process continues until the source node

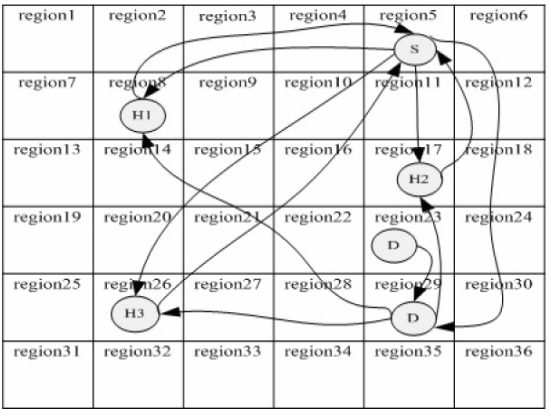


Fig. 1. Location Service of MHRLS

receives a reply message. This scheme uses less query/reply messages compared with the Query-All scheme, and it is highly efficient if all the home regions are not empty and all the home regions are reachable from any node in the network at any time. However, if these conditions do not hold, the query efficiency may be degraded, and the reliability may be reduced too.

4 Performance Analysis

In this section, we analyze the proposed MHRLS protocol in comparison with existing location service protocols in terms of scalability and reliability.

4.1 Scalability Analysis

We do the scalability analysis of our MHRLS protocol similar to the SLURP protocol in [9]. We define the scalability of a location service protocol as the cost to successfully update, maintain and query the location information. The total cost of a location service scheme can be divided into three parts: location update cost, location maintenance cost and location querying cost. In the following formulae, N stands for the number of mobile nodes in the network, and v stands for the moving speed of the mobile nodes. We derive all the formulae according to those in [9].

The location update cost of MHRLS c_u is:

$$c_u \propto kv\sqrt{N}\ ; \tag{1}$$

The location maintenance cost of MHRLS c_m is:

$$c_m \propto vN\ ; \tag{2}$$

The location querying cost of MHRLS c_q is:

$$c_q \propto k\sqrt{N}\ \text{(When the Query-All scheme is adopted);} \tag{3}$$

$$c_q \propto \sqrt{N} \text{ (When the Query-Nearest scheme is adopted);} \quad (4)$$

The total cost in the Query-All scheme c is:

$$c \propto (Nkv\sqrt{N} + vN + kN\sqrt{N}) \propto kvN^{3/2}; \quad (5)$$

The total cost in the Query-Nearest scheme is:

$$c \propto (kvN\sqrt{N} + vN + N\sqrt{N}) \propto kvN^{3/2}. \quad (6)$$

Thus, the total cost of MHRLS is: $c \propto kvN^{3/2}$. (7)

Table 1. Scalability Comparison between MHRLS and SLURP

	SLURP	MHRLS
location update cost	$c_u \propto v\sqrt{N}$	$c_u \propto kv\sqrt{N}$
location maintenance cost	$c_m \propto vN$	$c_m \propto vN$
location querying cost	$c_1 \propto \sqrt{N}$	$c_1 \propto k\sqrt{N}$ (the Query-All scheme)
		$c_1 \propto \sqrt{N}$ (the Query-Nearest scheme)
total cost	$c \propto vN^{3/2}$	$c \propto kvN^{3/2}$

Although Table 1 shows that the total cost of the MHRLS protocol is higher than that of the SLURP protocol, both of them scale at the same level when k is small. In fact, the scalability of the SLURP protocol is not as well as that shown in Table 1 because the scalability analysis of the SLURP protocol in [9] takes no account of the possible cost for maintaining location information when some region has no nodes within. Thus, we conclude that MHRLS has comparable scalability with SLURP.

4.2 Reliability Analysis

In this subsection, we compare the reliability of the proposed protocol to existing protocols in two different situations. In this paper, we define the reliability of a location service protocol as the probability to successfully update, maintain and query the location information in a certain situation.

4.2.1 Uniform Distribution of Empty Regions

Firstly, we assume that empty regions are of uniform distribution, i.e., the probability for each region in the network to be empty is the same. And we assume that the probability of any region to be empty is equal to p, which is very small. In the HALS protocol, since location information of each node is kept in a single region which is set to be the node's home region, the reliability of the protocol is equal to 1.0-p; In the SLURP protocol, since a copy of location information of each node is kept in 9 adjacent regions, the reliability of the protocol is 1.0-p^9; In the proposed MHRLS protocol, since a copy of location information of each node is kept in k uniformly distributed regions,

the reliability of the protocol is equal to 1.0-p^k, which is close to 1.0 if k is relatively large and p is very small. Therefore, the MHRLS protocol is more reliable than the HALS protocol in such situation. In addition, compared with SLURP, the MHRLS protocol achieves higher reliability if k is larger than 9; even if k is smaller than 9 (but not too small), reliability of the MHRLS protocol can be still very high even though it will be only a little lower than the SLURP protocol does. The analysis results on the reliability of the HALS, SLURP, and MHRLS protocols in the case of uniform distribution of empty regions are given in Table 2.

4.2.2 Non-uniform Distribution of Empty Regions

In this subsection, we consider that multiple adjacent regions become empty at the same time, for example, the 9 adjacent regions in the SLURP protocol are empty. In case of uniform distribution, such a case may not occur, or occurs very rarely. That is, such a case stands for an extreme case of non-uniform distribution of empty regions. This will probably happen when group mobility is of great importance, and thus a relatively large area may become empty. For example, a bombing in the battlefield may damage the region where it occurs, and it may also affect all its neighboring regions of the damaged region. Here we consider the probability that 9 adjacent regions are empty at the same time is equal to P. In this case, reliability of the HALS protocol is equal to 1.0-P, so is the SLURP protocol. However, in the same situation, the reliability of our proposed MHRLS protocol is equal to 1.0-P^k. That is because the k home regions of each node are evenly distributed in our protocol, and they have little probability that several of them happen to locate in the same 9 adjacent regions. Then we conclude that the MHRLS protocol is more reliable than both the HALS protocol and the SLURP protocol. The analysis results on the reliability of the HALS, SLURP, and MHRLS protocols in the case of non-uniform distribution of empty regions are also given in Table 2.

Table 2. Reliability Comparison with HALS and SLURP

	HALS	SLURP	MHRLS
Uniform Distribution of Empty Regions	1.0-p	1.0-p^9	1.0-p^k
Non-Uniform Distribution of Empty Regions	1.0-P	1.0-P	1.0-P^k

4.2.3 Numerical Results

According to Table 2, numerical results about the reliability of the HALS, SLURP and MHRLS protocols are given in Figure 2 and Figure 3 for uniform and non-uniform distributions of empty regions, respectively. In both figures, R stands for the reliability in Table 2, which is a function of the protocol being investigated and the probability for a given region to be empty. For the sake of presentation, we use the value of $-\lg(1.0-R)$ as Y. So the bigger the value of Y, the higher the R is. Therefore, in both Figure 2 and Figure 3, the X-axis essentially stands for the probability for a given region to be empty, and the Y-axis essentially stands for the reliability of the protocol being investigated.

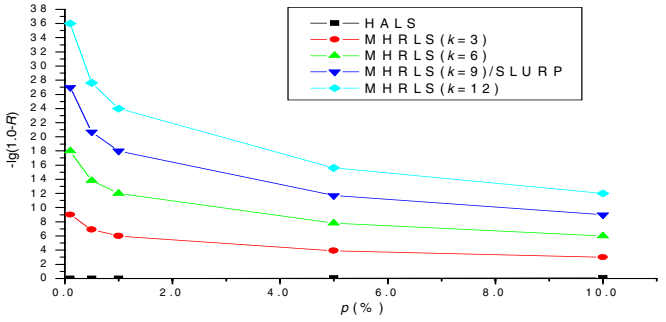


Fig. 2. Numerical Results in Uniform Distribution of Empty Regions

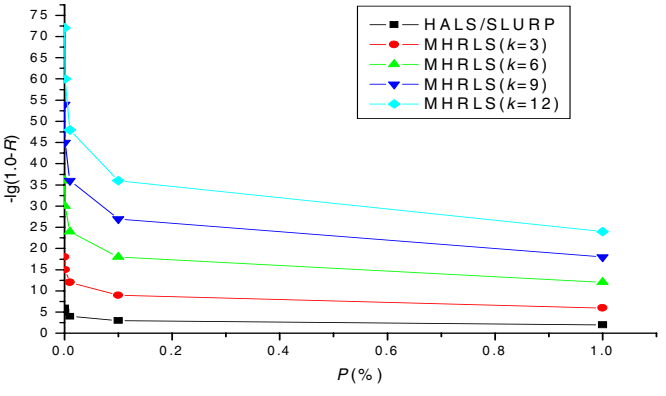


Fig. 3. Numerical Results in Non-Uniform Distribution of Empty Regions

Figure 2 shows that our proposed protocol always outperforms the HALS protocol under any probability for a given region to be empty, while our proposed protocol outperforms the SLURP protocol only when the parameter k is not less than 9. In addition, Figure 3 shows that our proposed protocol always outperforms both the HALS protocol and the SLURP protocol when the parameter k is larger than 1 (we only show k = 3, 6, 9, 12, respectively).

5 Conclusions

In this paper we proposed a scalable and reliable location service protocol in large-scale MANETs. The proposed protocol uses multiple home regions to update, maintain and query the location information for each node in the network. We also presented two kinds of query strategies which can be used in different application scenarios. Theoretical analysis shows the proposed protocol has comparative scalability as that of SLURP, and the proposed protocol is more reliable than HALS and SLURP in both uniform and non-uniform distributions of empty home regions.

Acknowledgments

This work is supported in part by the Hunan Provincial Natural Science Foundation of China No. 05JJ30118 (Secure group communications in large-scale mobile ad-hoc networks), and in part by the research grant of The Telecommunications Advancement Foundation of Japan.

References

1. M. Mauve, J. Widmer, and H. Hartenstein, "A Survey on Position-Based Routing in Mobile Ad-Hoc Networks," *IEEE Network*, Vol. 15, No. 6, pp. 30-39, November/December 2001.
2. T. Camp, J. Boleng, and L. Wilcox, "Location Information Services in Mobile Ad Hoc Networks," *Proc. IEEE ICC*, pp. 3318-3324, 2002.
3. T. Camp, J. Boleng, B. Williams, L. Wilcox, and W. Navidi, "Performance Evaluation of Two Location based Routing Protocols," *Proc. INFOCOM*, pp. 1678-1687, 2002
4. S. Capkun, M. Hamdi, and J.-P. Hubaux, "GPS-free Positioning in Mobile Ad Hoc Networks," *Proc. HICSS-34*, pp. 3481-3490, January 2001.
5. S. Basagni, I. Chlamtac, V.R. Syrotiuk, and B.A. Woodward, "A Distance Routing Effect Algorithm for Mobility (DREAM)," *Proc. ACM/IEEE MOBICOM*, pp. 76-84, 1998.
6. Z. J. Haas and B. Liang, "Ad Hoc Mobility Management with Uniform Quorum Systems," *IEEE/ACM Transactions on Networking*, Vol. 7, Issue 2, pp. 228-240, April 1999.
7. C.E. Perkins, "Mobile IP," *IEEE Communications Magazine*, Vol. 35, Issue 5, pp. 84-99, May 1997.
8. I. Stojmenovic, "Home Agent based Location Update and Destination Search Schemes in Ad Hoc Wireless Networks," *Computer Science, SITE, University of Ottawa, TR-99-10*, September 1999.
9. S.-C. Woo and S. Singh, "Scalable Routing Protocol for Ad Hoc Networks," *ACM Wireless Networks*, Vol. 7, Issue 5, pp. 513-529, 2001.
10. J. Li, J. Jannotti, D.S.J. De Couto, D.R. Karger, and R. Morris, "A Scalable Location Service for Geographic Ad Hoc Routing," *Proc. ACM/IEEE MOBICOM*, pp. 120-130, 2000.
11. C. T. Cheng, H. L. Lemberg, S.J.Philip, E. van den Berg, and T.Zhang, "SLALoM: A Scalable Location Mnagement Scheme for Large Mobile Ad-hoc Networks," *Proc. IEEE WCNC*, March 2002.
12. B.-C. Seet, Y. Pan, W.-J. Hsu,and C.-T. Lau, "Multi-Home Region Location Service for Wireless Ad Hoc Networks: An Adaptive Demand-driven Approach," *Proc. IEEE WONS*, pp. 258-263, 2005.
13. H. Takagi and L. Kleinrock, "Optimal Transmission Ranges for Randomly Distributed Packet Radio Terminals," *IEEE Transactions on Communications*, Vol. 32, No. 3, pp. 246-57, March 1984.

Global State Detection Based on Peer-to-Peer Interactions

Punit Chandra and Ajay D. Kshemkalyani

Computer Science Department, Univ. of Illinois at Chicago,
Chicago, IL 60607, USA
{pchandra, ajayk}@cs.uic.edu

Abstract. This paper presents an algorithm for global state detection based on peer-to-peer interactions. The interactions in distributed systems can be analyzed in terms of the peer-to-peer pairwise interactions of intervals between processes. This paper examines the problem: "If a global state of interest to an application is specified in terms of the pairwise interaction types between each pair of peer processes, how can such a global state be detected?" Devising an efficient algorithm is a challenge because of the overhead of having to track the intervals at different processes. We devise a distributed on-line algorithm to efficiently manage the distributed data structures and solve this problem. We prove the correctness of the algorithm and analyze its complexity.

1 Introduction

The pairwise interaction between processes is an important way of information exchange even in pervasive systems and large distributed systems such as peer-to-peer networks [12, 13] that do collaborative computing. We observe that the pairwise interactions of processes form a *basic building block* for information exchange. This paper advances the state-of-the-art in analyzing this building block by integrating it into the analysis of the dynamics of (i) global information exchange, and (ii) the resulting global states of a distributed system [5]. The study of global states and their observations, first elegantly formalized by Chandy and Lamport [5], is a fundamental problem in distributed computing [5, 9].

Many applications in a distributed peer-to-peer system inherently identify local durations or *intervals* at processes during which certain application-specific local predicates defined on local variables are true in a system execution [1]. Hence, we require a way to *specify* how durations at different processes are related to one another, and also a way to *detect* whether the specified relationships hold in an execution. The formalism and axiom system formulated in [7] identified a complete *orthogonal* set $\Re$ of 40 causality-based fine-grained temporal interactions (or relationships) between a pair of intervals to *specify* how durations at two peer processes are related to one another. The following problem **DOOR** for the Detection of Orthogonal Relations was formulated and addressed in [1]. *"Given a relation $r_{i,j}$ from $\Re$ for each pair of processes i and j, devise*

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 560–571, 2005.
© IFIP International Federation for Information Processing 2005

Table 1. Space, message and time complexities. Note: M = maximum queue length at P_0, the central server. $p \geq M$, as all the intervals may not be sent to P_0.

Centralized algorithm	Average time complexity at P_0	Total number of messages	Space at P_0 (= total message space)	Average space at P_i, $i \in [1, n]$
Fine_Rel [3, 4]	$O(n^2M)$ or $O(n[min(4m, np)])$	$O(min(4m, np))$	$O(min[(4n-2)np, 10nm])$	$O(n)$

Distributed Algorithms	Average time complexity/proc.	Total number of messages	Total average message space	Total space
Algorithm [1]	$O(min(np, 4mn))$	$O(n \cdot min(np, 4mn))$	$O(n^2 \cdot min(np, 4mn))$	$O(min(2np(2n-1), 10n^2m))$
this algorithm	$O(min(np, 4mn))$	$O(min(np, 4mn))$	$O(n^2 \cdot min(np, 4mn))$	$O(min(2np(2n-1), 10n^2m))$

a distributed on-line algorithm to identify the intervals, if they exist, one from each process, such that each relation $r_{i,j}$ is satisfied by the (i, j) process pair."

A solution satisfying the set of relations $\{r_{i,j}(\forall i, j)\}$ identifies a global state of the system [5]. Thus, the problem can be viewed as one of detecting a global state that satisfies the specified interval-based conditions per pair of peers.

Devising an efficient on-line algorithm to solve problem DOOR is a challenge because of the overhead of having to track the intervals at different processes. A distributed on-line algorithm to solve this problem was outlined in [1] without any formal discussion, without any analysis of its theoretical basis, and without any correctness proofs. A centralized but on-line algorithm was given in [3, 4]. In this paper, we devise a more efficient distributed on-line algorithm to solve this problem, and then prove its correctness. The algorithm uses $O(\min(np, 4mn))$ number of messages, where n is the number of processes, m is the maximum number of messages sent by any process, and p is the maximum number of intervals at any process. The total space complexity across all the processes is $\min(4n^2p - 2np, 10n^2m)$, and the average time complexity at a process is $O(\min(np, 4mn))$. The performance of the centralized algorithm [3, 4] and the algorithm in [1] are compared with the performance of the algorithm in this paper, in Table 1. The proposed algorithm uses an order of magnitude $O(n)$ messages fewer than the earlier algorithm [1], although that comes at the cost of somewhat larger messages.

2 System Model and Preliminaries

We assume an asynchronous distributed peer-to-peer system in which n processes communicate solely by reliable message passing over logical FIFO channels. $(E, \prec)$, where $\prec$ is an irreflexive partial ordering representing the causality or the "happens before" relation [10] on the event set E, is used as the model for a distributed system execution. E is partitioned into local executions at each process. Each E_i is a linearly ordered set of events executed by process P_i. We use N to denote the set of all processes.

We assume vector clocks [6, 11]. The durations of interest at each process can be the durations during which some local predicate of interest is true. Such a

Table 2. Dependent relations for interactions between intervals are given in the first two columns [7]. Tests for the relations are given in the third column.

Relation r	Expression for $r(X,Y)$	Test for $r(X,Y)$
R1	$\forall x \in X \forall y \in Y, x \prec y$	$V_y^-[x] > V_x^+[x]$
R2	$\forall x \in X \exists y \in Y, x \prec y$	$V_y^+[x] > V_x^+[x]$
R3	$\exists x \in X \forall y \in Y, x \prec y$	$V_y^-[x] > V_x^-[x]$
R4	$\exists x \in X \exists y \in Y, x \prec y$	$V_y^+[x] > V_x^-[x]$
S1	$\exists x \in X \forall y \in Y, x \not\preceq y \bigwedge y \not\preceq x$	$\exists x^0 \in X$: $V_y^-[y] \not\leq V_x^{x^0}[y] \wedge V_x^{x^0}[x] \not\leq V_y^+[x]$
S2	$\exists x_1, x_2 \in X \exists y \in Y, x_1 \prec y \prec x_2$	$\exists y^0 \in Y : V_x^+[y] \not< V_y^{y^0}[y] \wedge V_y^{y^0}[x] \not< V_x^-[x]$

Table 3. The 40 orthogonal relations in $\Re$ [7]. The upper part gives the 29 relations assuming dense time. The lower part gives 11 additional relations for nondense time.

Interaction Type	Relation $r(X,Y)$						Relation $r(Y,X)$					
	R1	R2	R3	R4	S1	S2	R1	R2	R3	R4	S1	S2
$IA(=IQ^{-1})$	1	1	1	1	0	0	0	0	0	0	0	0
$IB(=IR^{-1})$	0	1	1	1	0	0	0	0	0	0	0	0
$IC(=IV^{-1})$	0	0	1	1	1	0	0	0	0	0	0	0
$ID(=IX^{-1})$	0	0	1	1	1	1	0	1	0	1	0	0
$ID'(=IU^{-1})$	0	0	1	1	0	1	0	1	0	1	0	1
$IE(=IW^{-1})$	0	0	1	1	1	1	0	0	0	1	0	0
$IE'(=IT^{-1})$	0	0	1	1	0	1	0	0	0	1	0	1
$IF(=IS^{-1})$	0	1	1	1	0	1	0	0	0	1	0	1
$IG(=IG^{-1})$	0	0	0	0	1	0	0	0	0	0	1	0
$IH(=IK^{-1})$	0	0	0	1	1	0	0	0	0	0	1	0
$II(=IJ^{-1})$	0	1	0	1	0	0	0	0	0	0	1	0
$IL(=IO^{-1})$	0	0	0	1	1	1	0	1	0	1	0	0
$IL'(=IP^{-1})$	0	0	0	1	0	1	0	1	0	1	0	1
$IM(=IM^{-1})$	0	0	0	1	1	0	0	0	0	1	1	0
$IN(=IM'^{-1})$	0	0	0	1	1	1	0	0	0	1	0	0
$IN'(=IN'^{-1})$	0	0	0	1	0	1	0	0	0	1	0	1
$ID''(=(IUX)^{-1})$	0	0	1	1	0	1	0	1	0	1	0	0
$IE''(=(ITW)^{-1})$	0	0	1	1	0	1	0	0	0	1	0	0
$IL''(=(IOP)^{-1})$	0	0	0	1	0	1	0	1	0	1	0	0
$IM''(=(IMN)^{-1})$	0	0	0	1	0	0	0	0	0	1	1	0
$IN''(=(IMN')^{-1})$	0	0	0	1	0	1	0	0	0	1	0	0
$IMN''(=(IMN'')^{-1})$	0	0	0	1	0	0	0	0	0	1	0	0

duration, also termed as an *interval*, at process P_i is identified by the corresponding events within E_i. Each interval can be viewed as defining an event of higher granularity at that process [8], as far as the local predicate is concerned. Such higher-level events, one from each process, can be used to identify a global state. Intervals are denoted using X and Y. An interval X at P_i is denoted by X_i.

It was shown in [7] that there are 29 or 40 causality-based mutually orthogonal ways in which any two durations can be related to each other, depending on whether dense or nondense time is assumed. These orthogonal interaction types were identified by using the six relations given in the first two columns of Table 2. Relations R1 (strong precedence), R2 (partially strong precedence), R3 (partially weak precedence), R4 (weak precedence) define *causality conditions*. S1 and S2 define *coupling conditions*. The set of 40 relations is denoted as $\Re$.

Given a set of orthogonal relations, one between each pair of processes, that need to be detected, each of the 29 (40) possible independent relations in the dense (nondense) model of time can be tested for using the bit-patterns for the dependent relations, as given in Table 3 [7]. The tests for the relations $R1$ – $R4$, $S1$, and $S2$ using vector timestamps are given in the third column of Table 2. V_i^- and V_i^+ denote the vector timestamp at process P_i at the start and at the end of an interval, respectively. V_i^x denotes the vector timestamp of event x_i at process P_i. When the process is not specified explicitly, we assume that interval X occurs at P_i and interval Y occurs at P_j. For any two intervals X and X' that occur at the same process, if $R1(X, X')$, then we say that X is a *predecessor* of X' and X' is a *successor* of X.

Our goal is to efficiently apply the tests in Table 2 in a distributed manner across all the processes. Each process P_i, $1 \leq i \leq n$, maintains information about the timestamps of the start and end of its local intervals, and certain other local information, in a local queue Q_i. The n processes collectively run some distributed algorithm to process the information in the local queues and solve problem **DOOR**. In order for distributed algorithms to process the queued intervals efficiently, we first give some results about when two given intervals may potentially satisfy a given interaction type we want to detect.

3 Conditions for Satisfying Given Interaction Types

The discussion in this section is based on [1, 3, 4] which gave other (less efficient) solutions to solve problem DOOR.

Definition 1. Prohibition function $\mathcal{H} : \Re \rightarrow 2^{\Re}$ *is defined as* $\mathcal{H}(r_{i,j}) = \{R \in \Re \mid if\ R(X,Y)\ is\ true\ then\ r_{i,j}(X,Y')\ is\ false\ for\ all\ Y'\ that\ succeed\ Y\}$.

Definition 2. *The "allows" relation* $\rightsquigarrow$ *is a relation on* $\Re \times \Re$ *such that* $R' \rightsquigarrow R''$ *if the following holds. If* $R'(X,Y)$ *is true then* $R''(X,Y')$ *can be true for some* Y' *that succeeds* Y.

Lemma 1. *If* $R \in \mathcal{H}(r_{i,j})$ *then* $R \not\rightsquigarrow r_{i,j}$ *else if* $R \notin \mathcal{H}(r_{i,j})$ *then* $R \rightsquigarrow r_{i,j}$.

Proof: If $R \in \mathcal{H}(r_{i,j})$, using Definition 1, it can be inferred that $r_{i,j}$ is false for all Y' that succeed Y. This does not satisfy Definition 2. Hence $R \not\rightsquigarrow r_{i,j}$. If $R \notin \mathcal{H}(r_{i,j})$, it follows that $r_{i,j}$ can be true for some Y' that succeeds Y. This satisfies Definition 2 and hence $R \rightsquigarrow r_{i,j}$. □

Table 4 gives $\mathcal{H}(r_{i,j})$ for each of the 40 interaction types in $\Re$.

Theorem 1. *For* $R', R'' \in \Re$ *and* $R' \neq R''$, *if* $R' \rightsquigarrow R''$ *then* $R'^{-1} \not\rightsquigarrow R''^{-1}$.

Lemma 2. *If the relationship* $R(X,Y)$ *between intervals* X *and* Y *(belonging to process* P_i *and* P_j, *resp.) is contained in the set* $\mathcal{H}(r_{i,j})$, *and* $r_{i,j} \neq R$, *then interval* X *can be removed from the queue* Q_i.

Proof: From the definition of $\mathcal{H}(r_{i,j})$, we get that $r_{i,j}(X,Y')$ cannot exist, where Y' is any successor interval of Y. Further, $r_{i,j} \neq R$. So we have that X can never be a part of a solution and can be deleted from the queue. □

Table 4. $\mathcal{H}(r_{i,j})$ for the 40 orthogonal relations in $\Re$. The upper part is for 29 relations under dense time. The lower part is for 11 additional relations under nondense time.

Interaction Type $r_{i,j}$	$\mathcal{H}(r_{i,j})$	$\mathcal{H}(r_{j,i})$
$IA\ (=IQ^{-1})$	ϕ	$\Re - \{IQ\}$
$IB\ (=IR^{-1})$	$\{IA, IB, IF, II, IP, IO, IU, IX, IUX, IOP\}$	$\Re - \{IQ\}$
$IC\ (=IV^{-1})$	$\{IA, IB, IF, II, IP, IO, IU, IX, IUX, IOP\}$	$\Re - \{IQ\}$
$ID\ (=IX^{-1})$	$\Re - \{IQ, IS, IR, IJ, IL, IL', IL'', ID, ID', ID''\}$	$\Re - \{IQ\}$
$ID'\ (=IU^{-1})$	$\Re - \{IQ, IS, IR, IJ, IL, IL', IL'', ID, ID', ID''\}$	$\Re - \{IQ\}$
$IE\ (=IW^{-1})$	$\Re - \{IQ, IS, IR, IJ, IL, IL', IL'', ID, ID', ID''\}$	$\Re - \{IQ\}$
$IE'\ (=IT^{-1})$	$\Re - \{IQ, IS, IR, IJ, IL, IL', IL'', ID, ID', ID''\}$	$\Re - \{IQ\}$
$IF\ (=IS^{-1})$	$\Re - \{IQ, IS, IR, IJ, IL, IL', IL'', ID, ID', ID''\}$	$\Re - \{IQ\}$
$IG\ (=IG^{-1})$	$\Re - \{IQ, IR, IJ, IV, IK, IG\}$	$\Re - \{IQ, IR, IJ, IV, IK, IG\}$
$IH\ (=IK^{-1})$	$\Re - \{IQ, IR, IJ, IV, IK, IG\}$	$\Re - \{IQ, IR, IJ\}$
$II\ (=IJ^{-1})$	$\Re - \{IQ, IR, IJ, IV, IK, IG\}$	$\Re - \{IQ, IR, IJ\}$
$IL\ (=IO^{-1})$	$\Re - \{IQ, IR, IJ\}$	$\Re - \{IQ, IR, IJ\}$
$IL'\ (=IP^{-1})$	$\Re - \{IQ, IR, IJ\}$	$\Re - \{IQ, IR, IJ\}$
$IM\ (=IM^{-1})$	$\Re - \{IQ, IR, IJ\}$	$\Re - \{IQ, IR, IJ\}$
$IN\ (=IM'^{-1})$	$\Re - \{IQ, IR, IJ\}$	$\Re - \{IQ, IR, IJ\}$
$IN'\ (=IN'^{-1})$	$\Re - \{IQ, IR, IJ\}$	$\Re - \{IQ, IR, IJ\}$
$ID''\ (=(IUX)^{-1})$	$\Re - \{IQ, IS, IR, IJ, IL, IL', IL'', ID, ID', ID''\}$	$\Re - \{IQ\}$
$IE''\ (=(ITW)^{-1})$	$\Re - \{IQ, IS, IR, IJ, IL, IL', IL'', ID, ID', ID''\}$	$\Re - \{IQ\}$
$IL''\ (=(IOP)^{-1})$	$\Re - \{IQ, IR, IJ\}$	$\Re - \{IQ, IR, IJ\}$
$IM''\ (=(IMN)^{-1})$	$\Re - \{IQ, IR, IJ\}$	$\Re - \{IQ, IR, IJ\}$
$IN''\ (=(IMN')^{-1})$	$\Re - \{IQ, IR, IJ\}$	$\Re - \{IQ, IR, IJ\}$
$IMN''\ (=(IMN'')^{-1})$	$\Re - \{IQ, IR, IJ\}$	$\Re - \{IQ, IR, IJ\}$

Lemma 3. *If the relationship between a pair of intervals X and Y (belonging to processes P_i and P_j respectively) is not equal to $r_{i,j}$, then interval X or interval Y is removed from its queue Q_i or Q_j, respectively.*

Proof: We use contradiction. Assume relation $R(X,Y)$ $(\neq r_{i,j}(X,Y))$ is true for intervals X and Y. From Lemma 2, the only time neither X nor Y will be deleted is when $R \notin \mathcal{H}(r_{i,j})$ and $R^{-1} \notin \mathcal{H}(r_{j,i})$. From Lemma 1, it can be inferred that $R \rightsquigarrow r_{i,j}$ and $R^{-1} \rightsquigarrow r_{j,i}$. As $r_{i,j}^{-1} = r_{j,i}$, we get $R \rightsquigarrow r_{i,j}$ and $R^{-1} \rightsquigarrow r_{i,j}^{-1}$. This is a contradiction as by Theorem 1, R being unequal to $r_{i,j}$, $R \rightsquigarrow r_{i,j} \Rightarrow R^{-1} \not\rightsquigarrow r_{i,j}^{-1}$. Hence $R \in \mathcal{H}(r_{i,j})$ or $R^{-1} \in \mathcal{H}(r_{j,i})$, and thus at least one of the intervals will be deleted.

4 A Distributed Peer-to-Peer Algorithm

Each process P_i, where $1 \leq i \leq n$, maintains the following data structures. (1) V_i : array[1..n] of integer. This is the *Vector Clock* [6, 11]. (2) I_i : array[1..n] of integer. This is a *Interval Clock* [1, 3, 4] which tracks the latest intervals at

1. When an internal event or send event occurs at process P_i, $V_i[i] = V_i[i] + 1$.
2. Every message contains the vector clock and *Interval Clock* of its send event.
3. When process P_i receives a message msg, then $\forall\ j$ do,
   if $(j == i)$ **then** $V_i[i] = V_i[i] + 1$,
   else $V_i[j] = \max(V_i[j], msg.V[j])$.
4. When an interval starts at P_i (local predicate ϕ_i becomes true), $I_i[i] = V_i[i]$.
5. When process P_i receives a message msg, then $\forall\ j$ do,
   $I_i[j] = \max(I_i[j], msg.I[j])$

Fig. 1. The protocol for maintaining vector clock and *Interval Clock*

```
type Event_Interval = record
    interval_id : integer;
    local_event: integer;
end

type Log = record
    start: array[1..n] of integer;
    end: array[1..n] of integer;
    p_log: array[1..n] of Process_Log;
end

type Process_Log = record
    event_interval_queue: queue of Event_Interval;
end
```

Start of an interval:

$Log_i.start = V_i^-$.

On receiving a message during an interval:

if (change in I_i) **then**
  for each k such that $I_i[k]$ was changed
    insert $(I_i[k], V_i[i])$ in $Log_i.p_log[k].event_interval_queue$

End of interval:

$Log_i.end = V_i^+$

if (a receive or send occurs between start of previous and end of present interval) **then**
  Enqueue Log_i on to the local queue Q_i.

Fig. 2. The data structures and the protocol for constructing Log at P_i $(1 \leq i \leq n)$

For $S2(X, Y)$:

1. (1a) **for** each $event_interval \in Log_j.p_log[i].event_interval_queue$
   (1b) **if** $(event_interval.interval_id < Log_i.start[i])$
   (1c) **then** remove $event_interval$
2. (2a) $temp = \infty$
   (2b) **if** $(Log_j.start[i] \geq Log_i.start[i])$ **then** $temp = Log_j.start[j]$
   (2c) **else for** each $event_interval \in Log_j.p_log[i].event_interval_queue$
   (2d) $temp = \min(temp, event_interval.local_event)$
3. **if** $(Log_i.end[j] \geq temp)$ **then** $S2(X, Y)$ is true.

For $S1(Y, X)$:

1. Same as step 1 of scheme to determine $S2(X, Y)$.
2. Same as step 2 of scheme to determine $S2(X, Y)$.
3. **if** $(Log_i.end[j] < temp)$ and $(temp > Log_j.start[j])$ **then** $S1(Y, X)$ is true.

Fig. 3. The protocol to test for $S1(X, Y)$ and $S2(Y, X)$

processes. $I_i[j]$ is the timestamp $V_j[j]$ when the predicate ϕ_j of interest at P_j last became true, as known to P_i. (3) Log_i: contains the information about an interval, needed to compare it with other intervals. Fig. 1 shows how to update the vector clock and *Interval Clock*.

Log_i is constructed and stored on the local queue Q_i using the data structures and protocol shown in Fig. 2. The Log is used to determine the relationship between two intervals. The tests in Table 2 are used to find which of $R1$ – $R4$, $S1$, and $S2$ are true. Fig. 3 shows how to implement the tests for $S1$ and $S2$. The data structures in Fig. 2, 3 were proposed and used in the design of the previous algorithms [1, 3, 4] to address problem DOOR. However, the algorithm in this paper is fully distributed and more efficient.

The algorithm identifies a set of intervals $\mathcal{I}$, if they exist, one interval I_i from each process P_i, such that the relation $r_{i,j}(I_i, I_j)$ is satisfied by each (i, j) process pair. If no such set of intervals exists, the algorithm does not return any interval set. The algorithm uses a token T. The data structure for the token (T) is given in Figure 4. $T.log[i]$ contains the Log corresponding to the interval at the head of

type T = **token**
log: array [1..n] of *Log*;
//Contains the *Log*s of the intervals (at the queue heads) which may be in soln.
C: array [1..n] of boolean;
//$C[i]$ is true if and only if $Log[i]$ at the head of Q_i can be a part of soln.
end

(1) Initial state for process P_i
(1a) Q_i has a dummy interval

(2) Initial state for the token
(2a) $\forall i : T.C[i] = false$
(2b) T does not contain any *Log*
(2c) A randomly elected process P_i holds the token

(3) On receiving token T at P_i
(3a) **while** $(T.C[i] = false)$
(3b) Delete head of the queue Q_i
(3c) **if** (Q_i is empty) **then** wait until Q_i is non-empty
(3d) $T.C[i] = true$
(3e) X = head of Q_i
(3f) **for** $j = 1$ to n
(3g) **if** $(T.C[j] = true)$ **then**
(3h) $Y = T.log[j]$
(3i) Determine $R(X, Y)$ using the tests given in Fig. 3 and Table 2
(3j) **if** $(r_{i,j} \neq R(X, Y))$ and $(R(X, Y) \in \mathcal{H}(r_{i,j}))$ **then** $T.C[i] = false$
(3k) **if** $(r_{j,i} \neq R(Y, X))$ and $(R(Y, X) \in \mathcal{H}(r_{j,i}))$ **then**
(3l) $T.C[j] = false$
(3m) $T.log[j] = \perp$
(3n) $T.log[i] = Log_i$
(3o) **if** $(\forall k : T.C[k] = true)$ **then**
(3p) solution found. T has the solution *Log*s.
(3q) **else**
(3r) $k = i + 1$
(3s) **while** $(T.C[k] \neq false)$
(3t) $k = (k + 1)\ mod\ n$
(3u) Send T to P_k

Fig. 4. Distributed algorithm to solve problem DOOR

queue Q_i. $T.C[i] = true$ implies that the interval at the head of queue Q_i may be a part of the final solution and the corresponding log Log_i is stored in the token. If $T.C[i] = false$ then the interval at the head of queue Q_i is not a part of the solution, its corresponding log is not contained in the token, and the interval can be deleted. The algorithm is given in Figure 4. The process (P_i) receives a token only if $T.C[i] = false$, which means the interval at the head of queue Q_i is not a part of the solution and hence is deleted. The next interval X on the queue Q_i is then compared with each other interval Y whose log Log_j is contained in $T.log[j]$ (in which case $T.C[j] = true$, lines 3e-3i). According to Lemma 3,

the comparison between intervals X and Y can result in three cases. (1) $r_{i,j}$ is satisfied. (2) $r_{i,j}$ is not satisfied and interval X can be removed from the queue Q_i. (3) $r_{i,j}$ is not satisfied and interval Y can be removed from the queue Q_j. In the third case, the log Log_j corresponding to interval Y is deleted and $T.C[j]$ is set to false (lines 3l-3m). In the second case, $T.C[i]$ is set to false (line 3j) so that in the next iteration of the **while** loop, the interval X is deleted (lines 3a-3b). Note that both cases (2) and (3) can be true as a result of a comparison. The above process is repeated until the interval at the head of the queue Q_i satisfies the required relationships with each of the interval *Log*s remaining in the token (T). The process (P_i) then adds the log Log_i corresponding to the interval at the head of queue Q_i to the token $T.Log[i]$ and sets $T.C[i]$ equal to true. A solution is detected when $T.C[k]$ is true for all indices k (lines 3n-3p), and is given by all the n log entries of all the processes, $T.Log[1, \ldots, n]$. If the above condition (line 3o) is not satisfied then the token is sent to some process P_j whose log Log_j is not contained in the token $T.Log[j]$ (in which case $T.C[j] = false$, lines 3r-3u). Note that the wait in (line 3c) can be made non-blocking by restructuring the code using an interrupt-based approach.

5 Correctness Proof

Lemma 4. *After P_i executes the loop in lines (3f-3m), if $T.C[i] = true$ then the relationship $r_{i,j}$ is satisfied for interval X_i at the head of queue Q_i and each interval Y_j at the head of queue Q_j satisfying $T.C[j] = true$.*

Proof: The body of the loop (lines 3j-l) implements Lemma 2 by testing for $R(X_i, Y_j) \in \mathcal{H}(r_{i,j})$ and $R(Y_j, X_i) \in \mathcal{H}(r_{j,i})$, when R is not equal to $r_{i,j}$. If $r_{i,j}$ is not satisfied between interval X and interval Y then by Lemma 3, X or Y is deleted, i.e., (line 3j) or (lines 3k, 3l) is executed and hence $T.C[i]$ or $T.C[j]$ is set to false. This implies if both $T.C[i]$ and $T.C[j]$ are true then the relationship $r_{i,j}(X, Y)$ is true.

It remains to show that for all j for which $T.C[j]$ is true when the loop in (3f-3m) completes, Y_j which is in $T.Log[j]$ is the same as $head(Q_j)$. This follows by observing that (i) $T.Log[j]$ was the same as $head(Q_j)$ when the token last visited and left P_j, and (ii) $head(Q_j)$ is deleted only when $T.C[j]$ is false and hence the token visits P_j. □

Theorem 2. *When a solution $\mathcal{I}$ is detected by the algorithm in Figure 4, the solution is correct, i.e., for each $i, j \in N$ and $I_i, I_j \in \mathcal{I}$, the intervals $I_i = head(Q_i)$ and $I_j = head(Q_j)$ are such that $r_{i,j}(I_i, I_j)$.*

Proof: It is sufficient to prove that for the solution detected, which happens at the time $T.C[k] = true$ for all k (lines 3o,p), (i) $r_{i,j}(I_i, I_j)$ is satisfied for all pairs (i, j), and (ii) none of the queues is empty. To prove (i) and (ii), note that at this time, the token must have visited each process at least once because only the token-holder P_i can set $T.C[i]$ to true. Consider the latest time t_i when process P_i was last visited by the token (and $T.C[i]$ was set to true and $T.Log[i]$ was set to

$head(Q_i)$). Since t_i until the solution is detected, $T.C[i]$ remains true, otherwise the token would revisit P_i again (lines 3s-u) – leading to a contradiction. Linearly order the process indices in array $Visit[1, \ldots, n]$ according to the increasing order of the times of the last visit of the token. Then for k from 2 to n, we have that when the token was at $P_{Visit[k]}$, the intervals corresponding to $T.Log[k]$ and $T.Log[m]$, for all $1 \leq m < k$, were tested successfully for relation $r_{k,m}$ and $T.C[k]$ and $T.C[m]$ were true after this test. This shows that the intervals from every pair of processes got tested, and by Lemma 4, that $r_{k,m}(X_k, Y_m)$ was satisfied for $X_k = head(Q_k)$ and $Y_m = head(Q_m)$ at the time of comparison.

As shown above, since t_k until the solution is detected, $T.C[k]$ remains true and the token does not revisit P_k. Hence, from t_k until the solution is detected, none of the intervals tested at t_k using $T.Log$ got dequeued from their respective queues and $r_{k,m}(X_k, Y_m)$ continues to be satisfied for $X_k = head(Q_k)$ and $Y_m = head(Q_m)$ when the solution is detected. □

Let $\mathcal{I}(h)$ denote the set of intervals at the head of each queue, that are compared during the processing triggered by hop h of the token. Each $\mathcal{I}(h)$ identifies a system state (not necessarily consistent). Observe that for any $\mathcal{I}(h)$ and $\mathcal{I}(h+1)$ and any P_i, interval $I_i(h+1)$ in $\mathcal{I}(h+1)$ is equal to or an immediate successor of interval $I_i(h)$ in $\mathcal{I}(h)$. We thus say that all the $\mathcal{I}$ are linearly ordered, and $\mathcal{I}(h)$ *precedes* $\mathcal{I}(h')$, for all $h' > h$. Let $\mathcal{I}(S)$ denote the set of intervals that form the *first* solution, assuming it exists. Let $\mathcal{I}(b)$ denote the first value of $\mathcal{I}(h)$ that contains one (or more) of the intervals belonging to $\mathcal{I}(S)$. Let $\mathcal{I}(init)$ denote the initial set of intervals. Clearly, $\mathcal{I}(init)$ precedes $\mathcal{I}(b)$ which precedes $\mathcal{I}(S)$.

Lemma 5. *In any hop h of the token, no interval $X_i \in \mathcal{I}(S)$ gets deleted.*

Proof: This can be shown by considering two possibilities.

1. An interval X_i in $\mathcal{I}(S)$ get compared with some interval Y_j that is also in $\mathcal{I}(S)$. In this case, both conditions are false in lines 3j and 3k as $r_{i,j}(X_i, Y_j)$ is satisfied, and *this comparison* does not cause either of the intervals to be deleted from $T.Log[]$. Also, $T.C[i]$ and $T.C[j]$ remain true.
2. An interval X_i gets compared with some $Y_j \in \mathcal{I}(h) \setminus \mathcal{I}(S)$. Observe that Y_j is a predecessor of the interval Y_j' at P_j such that $Y_j' \in \mathcal{I}(S)$ and thus $r_{i,j}(X_i, Y_j')$ holds. We have that $R(X_i, Y_j) \neq r_{i,j}$. We now show that Y_j gets deleted and X_i does not get deleted.
   - Applying Theorem 1, we have $R(X_i, Y_j) \rightsquigarrow r_{i,j}(X_i, Y_j')$ which implies $R^{-1} \not\rightsquigarrow r_{j,i}$. From Lemma 1, we conclude that $R^{-1} \in \mathcal{H}(r_{j,i})$. There are two cases to consider. (a) The token is at P_i. By Lemma 2 which is implemented in lines 3k-m, the comparison results in Y_j being deleted from $T.Log$ and subsequently from Q_j (line 3b when the token reaches P_j). (b) The token is at P_j. By Lemma 2 which is implemented in lines 3j and 3a-3e, the comparison results in Y_j being deleted from Q_j.
   - As $R \rightsquigarrow r_{i,j}$ therefore from Lemma 1, we conclude that $R \notin \mathcal{H}(r_{i,j})$. By Lemma 2 which is implemented in lines 3j and 3k-m (depending on whether P_i or P_j has the token), this comparison does not result in X_i being deleted. □

Lemma 6. *In any hop h of the token, at least one interval $Y_j \in \mathcal{I}(h) \setminus \mathcal{I}(S)$ gets deleted.*

Proof: Line 3b deletes interval ($head(Q_i)$) when the token arrives at P_i. □

Theorem 3. *If a solution $\mathcal{I}$ exists, i.e., for each $i, j \in N$, the intervals I_i, I_j belonging to $\mathcal{I}$ are such that $r_{i,j}(I_i, I_j)$, then the solution is detected by the algorithm in Figure 4.*

Proof: From state $\mathcal{I}(init)$ in which $T.Log$ was initialized to contain no log, onwards until state $\mathcal{I}(b)$, Lemma 6 is true. Hence, in each hop of the token, the interval at the head of the queue of some process must get replaced by the immediate successor interval at that process. This guarantees progress and that state $\mathcal{I}(b)$ is reached.

From state $\mathcal{I}(b)$ onwards until state $\mathcal{I}(S)$, both Lemmas 5 and 6 apply. Lemma 6 guarantees progress (some interval not belonging to $\mathcal{I}(S)$ gets replaced by its immediate successor interval at every hop). Lemma 5 guarantees that no interval belonging to the solution set of $\mathcal{I}(S)$ gets deleted. Once $T.Log$ contains all the intervals of $\mathcal{I}(S)$ and hence $T.C[k]$ is true for all k, the solution is detected. □

Thus, Theorems 2 and 3 show that the algorithm detects a solution if and only if it exists.

6 Complexity Analysis

The complexity analysis sketched below and summarized in Table 1 is in terms of two parameters – the maximum number of messages sent per process (m) and the maximum number of intervals per process (p). Details are given in [2].

Space complexity at P_1 to P_n:

1. In terms of m: It is necessary to store *Log* for four intervals for every message – two for the send event and two for the receive event. Refer [2, 3, 4] for the reasoning. As there are a total of nm number of messages exchanged between all processes, a total of $4nm$ interval *Log*s are stored across all the queues, though not necessarily at the same time.
   - The *total space overhead* across all processes is $2.mn.n + 4mn.2n = 10mn^2$. The term $2.mn.n$ arises because for each of the mn messages sent, each of the other n processes eventually (due to transitive propagation of *Interval Clock*) may need to insert a *Event_Interval* tuple (size 2) in its *Log*. This can generate $2.mn.n$ overhead in *Log*s across the n processes. The term $4mn.2n$ arises because the vector timestamp at the start and at the end of each interval is also stored in each *Log*. It now follows that the average number of *Log*s per process is $4m$ and the average space overhead per process is $10mn$. Also note that the average size of a *Log* is $O(n)$.

- For a process, the *worst case* occurs when it receives m messages from each of the other $n-1$ processes. Assuming the process sends m messages, a total of $m(n-1)+m$ messages are sent or received by this process. In this case, the number of *Log*s stored on the process queue is $2mn$, two *Log*s for each receive event or send event (see [2, 3, 4] for justification). The total space required at the process is $2mn.2n+2m(n-1) = O(mn^2)$. The term $2mn.2n$ arises from the fact that each *Log* contains two vector timestamps and there are a total of $2mn$ *Log*s stored on the process queue. The term $2m(n-1)$ arises because an *Event_Interval* is added for each receive and there are a total of $m(n-1)$ messages received on the process.

Note that the worst case just discussed is for a single process; the total space overhead always remains $O(mn^2)$ and on an average, the space complexity for each process is $O(nm)$.

2. In terms of p: The total number of *Log*s stored at each process is p because in the worst case, the *Log* for each interval may need to be stored. The total number of *Log*s stored at all the processes is np. Consider the cumulative space requirement for *Log* over all the intervals at a single process.
   - Each *Log* stores the start (V^-) and the end (V^+) of an interval, which requires a maximum of $2np$ integers over all *Log*s per process.
   - Additionally, an *Event_Interval* is added to the *Log* for every component of *Interval Clock* which is modified due to the receive of a message. Since a change in a component of *Interval Clock* implies the start of a new interval on another process, the total number of times the component of *Interval Clock* can change is equal to the number of intervals on all the processes. Thus the total number of *Event_Interval* which can be added to the *Log* of a single process is $(n-1)p$. This gives a total of $2(n-1)p$ integers (corresponding to *Event_Interval*s) at each process.

   The total space required for *Log*s corresponding to all p intervals on a single process is $2(n-1)p + 2np$. So the total space is $4n^2p - 2np = O(n^2p)$.

Thus, the total number of *Log*s stored on all the processes is $\min(np, 4mn)$ and the total space overhead for all the processes is $\min(4n^2p - 2np, 10n^2m)$.

Time Complexity: The time complexity is the product of the number of steps required to determine an orthogonal relationship from $\Re$ between a pair of intervals, and the number of interval pairs considered.

- The first part of the product on average takes $O(1)$ time (to determine a relationship [2, 3, 4]). Note this part does not depend on the algorithm being centralized or distributed.
- To analyze the second part of the product, consider Figure 4. The maximum number of times an interval at the head of its queue is compared locally with intervals contained in the token is $(n-1)$. The reason being that when an interval comes to the head of a queue (Q_i), it may be compared with $n-1$ other intervals (contained in the token), one corresponding to each other process, but the next time that token reaches the process (P_i) is when $C[i] = false$ and hence the interval is dequeued. Thus the total number of

interval pairs compared is $(n-1)$ times the total number of *Log*s over all the queues. The total number of *Log*s over all the queues was shown above to be equal to $\min(np, 4mn)$, hence the total number of interval pairs compared is $(n-1)\min(np, 4mn)$.

As on average it takes $O(1)$ time to determine a relationship, the average time complexity of the algorithm is equal to $O(n \cdot \min(np, 4mn))$. Hence the average time complexity per process is $O(\min(np, 4mn))$.

Message Complexity: The token is sent to P_j whenever $C[j]$ is false, and $C[j]$ is false if the interval at the head of the queue Q_j has to be deleted. Thus, the maximum number of times the token is sent is equal to the total number of intervals across all the queues, which is equal to $\min(np, 4nm)$. Hence, the total number of messages sent is $\min(np, 4nm)$. The maximum number of *Log*s stored on a token in $n-1$ and the size of each *Log* on the average can be seen to be $O(n)$. Thus, the total message space overhead is $O(n^2 \min(np, 4mn))$.

References

1. P. Chandra, A. D. Kshemkalyani, Detection of orthogonal interval relations. Proc. 9^{th} Intl. High Performance Computing Conference (HiPC), LNCS 2552, Springer, 323-333, 2002.
2. P. Chandra, A. D. Kshemkalyani, Distributed detection of temporal interactions. Tech. Report UIC-ECE-02-07, Univ. of Illinois at Chicago, May 2002.
3. P. Chandra, A. D. Kshemkalyani, Detecting global predicates under fine-grained modalities. Proc. 8th Asian Computing Conference (ASIAN), LNCS 2896, Springer-Verlag, 91-109, December 2003.
4. P. Chandra, A. D. Kshemkalyani, Causality-based predicate detection across space and time. IEEE Transactions on Computers, 54(11): 1438-1453, November 2005.
5. K. M. Chandy, L. Lamport, Distributed snapshots: Determining global states of distributed systems. ACM Transactions on Computer Systems, 3(1): 63-75, 1985.
6. C. J. Fidge, Timestamps in message-passing systems that preserve partial ordering. Australian Computer Science Communications, 10(1): 56-66, February 1988.
7. A. D. Kshemkalyani, Temporal interactions of intervals in distributed systems. Journal of Computer and System Sciences, 52(2): 287-298, April 1996.
8. A. D. Kshemkalyani, A framework for viewing atomic events in distributed computations. Theoretical Computer Science, 196(1-2), 45-70, April 1998.
9. A. D. Kshemkalyani, M. Raynal, M. Singhal, An introduction to global snapshots of a distributed system. IEE/IOP Distributed Systems Engineering Journal, 2(4): 224-233, December 1995.
10. L. Lamport, Time, clocks, and the ordering of events in a distributed system. Communications of the ACM, 21(7): 558-565, July 1978.
11. F. Mattern, Virtual time and global states of distributed systems. Parallel and Distributed Algorithms, North-Holland, 215-226, 1989.
12. D. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, Z. Xu, Peer-to-peer computing. Hewlett Packard Technical Report HPL-2002-57, 2002.
13. J. Rissom, T. Moors, Survey of research towards robust peer-to-peer networks: Search methods. Technical Report UNSW-EE-P2P-1-1, Univ. of New South Wales, 2004.

Nonintrusive Snapshots Using Thin Slices

Ajay D. Kshemkalyani and Bin Wu

Computer Science Department, Univ. of Illinois at Chicago,
Chicago, IL 60607, USA
{ajayk, bwu}@cs.uic.edu

Abstract. This paper gives an efficient algorithm for recording consistent snapshots of an asynchronous distributed system execution. The nonintrusive algorithm requires $6(n-1)$ control messages, where n is the number of processes. The algorithm has the following properties. (P1) The application messages do not require any changes, not even the use of timestamps. (P2) The application program requires no changes, and in particular, no inhibition is required. (P3) Any process can initiate the snapshot. (P4) The algorithm does not use the message history. A simple and elegant three-phase strategy of uncoordinated observation of local states is used to give a consistent distributed snapshot. Two versions of the algorithm are presented. The first version records consistent process states without requiring FIFO channels. The second version records process states and channel states consistently but requires FIFO channels. The algorithm also gives an efficient way to detect any stable property, which was an unsolved problem under assumptions (P1)-(P4).

1 Problem Definition

A distributed system is modeled as a directed graph (N, L), where N is the set of processes and L is the set of links connecting the processes. Let $n = |N|$ and $l = |L|$. A distributed snapshot represents a consistent global state of the system. Recording distributed snapshots of an execution is a fundamental problem in asynchronous message-passing systems. Since the seminal algorithm of Chandy and Lamport [3] which is a non-inhibitory algorithm that requires FIFO channels and $2l$ messages to record a snapshot, plus additional messages to assemble the snapshot, several algorithms have been proposed. A survey is given in [8].

This paper gives an efficient nonintrusive non-inhibitory algorithm for recording consistent snapshots of an asynchronous distributed system execution. The algorithm requires $6(n-1)$ control messages and has the following properties.

P1. The application messages require no changes, not even timestamps.
P2. The application program requires no changes, implying no inhibition.
P3. Any process can initiate the snapshot.
P4. The algorithm does not require the log of the message history.

These properties are important for ubiquitous and pervasive computing. A simple and elegant three-phase strategy of uncoordinated observation of local

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 572–583, 2005.
© IFIP International Federation for Information Processing 2005

Table 1. Comparison of global snapshot algorithms. The acronym p.b. denotes that control information is piggybacked on the application messages.

Algorithm	Channels required	Non-inhibitory	Application messages unmodified	Number of control messages	Snapshot collection not needed	Message history not used
Chandy-Lamport [3]	FIFO	Y	Y	$O(n^2)$	N	Y
Spezialetti-Kearns [16]	FIFO	Y	Y	$O(n^2)$	N	Y
Venkatesan [17]	FIFO	Y	Y	$O(n^2)$	N	Y
Helary [5] (wave sync.)	FIFO	N	Y	$O(n^2)$	N	Y
Helary [5] (wave sync.)	non-FIFO	N	Y	$O(n^2)$	N	Y
Lai-Yang [10]	non-FIFO	Y	N (p.b.)	$O(n^2)$	N	N
LRV [12]	non-FIFO	Y	N (p.b.)	$O(n^2)$	N	N
Mattern [14]	non-FIFO	Y	N (p.b.)	$O(n)$	N	Y
Acharya-Badrinath [1]	CO	Y	Y	$2n$	Y	Y
Alagar-Venkatesan [2]	CO	Y	Y	$3n$	Y	Y
Proposed snapshot	FIFO	Y	Y	$6n$	Y	Y
Proposed snapshot (w/o channel states)	non-FIFO	Y	Y	$6n$	Y	Y

states gives a consistent distributed snapshot. Two versions of the algorithm are presented: the first version records consistent process states without requiring FIFO channels and without using any form of message send/receive or event counters, whereas the second version records process states and channel states consistently but requires FIFO channels. Critchlow and Taylor have shown that for a system with non-FIFO channels, a snapshot algorithm must either use piggybacking or use inhibition [4]. Hence, the second version of the algorithm cannot be improved upon to also record channel states while retaining the properties of no inhibition and no piggybacking while using non-FIFO channels.

Table 1 compares the proposed algorithms with the existing algorithms. Besides serving as a general-purpose snapshot algorithm, the proposed algorithm can detect *any* arbitrary stable predicate [3], which was an unsolved problem under the assumptions P1-P4. While many specialized algorithms are tailored to specific stable properties, such as deadlock, termination, and garbage, the following algorithms detect general stable predicates.

- The Marzullo-Sabel algorithm [13] can detect only *locally stable* predicates.
- The Schiper-Sandoz algorithm [15] can detect only *strong stable* predicates.
- Kshemkalyani-Singhal's two-phase algorithm [9], based on Ho-Ramamoorthy's algorithm [6], showed how to correctly detect deadlocks. A general method to detect stable properties was then outlined [9]. In essence, if a property does not change between the two serial phases of uncoordinated observations, the property must hold at some instant between the two phases. If it is stable, it must hold henceforth. While locally stable predicates can be detected satisfying assumptions (P1)-(P4) and without assuming FIFO channels, details of detecting arbitrary stable predicates are not given.

Neither Marzullo and Sabel [13] nor Schiper and Sandoz [15] showed any relationship between the classes of strong stable and locally stable properties. These existing algorithms can only detect some subclass of stable predicates, and do not satisfy (P1)-(P4). The proposed algorithm can detect *any* stable predicate.

Summary of Main Contributions:

1. The snapshot algorithms we propose for FIFO channels and for non-FIFO channels are linear in the number of messages, and satisfy (P1)-(P4).
2. The non-FIFO version of our snapshot algorithm can be used to detect locally stable predicates, under assumptions (P1)-(P4).
3. The FIFO version of our snapshot algorithm can be used to detect *any* stable predicate, under assumptions (P1) to (P4).

System Model:
An asynchronous execution in a distributed system is modeled as $(E, \prec)$, where $\prec$ is the causality relation [11] on the set of events E in the execution. $E = \bigcup_{i \in N} E_i$, where E_i is the totally ordered chain of event at process P_i. An event executed by P_i is denoted e_i. A *cut* C is a subset of E such that the events of a cut are downward-closed within each E_i. A *consistent cut* is a downward-closed subset of E. The system state after the events in a cut is a global state; if the cut is consistent, the corresponding system state is termed a *consistent global state*. An *execution slice* is defined as the difference of two cuts $D \setminus C$, where $C \subseteq D$. The slice is also referred to as a *suffix* of C with respect to D. When D is not explicitly specified, its default value is the execution E.

The execution history at a process P_i is a sequence of alternating states and local events, $\langle s_i^0, e_i^1, s_i^1, e_i^2, s_i^2, e_i^3, s_i^3, \ldots \rangle$. Events and messages of the snapshot algorithm form a superimposed control execution. Among the application events and messages, those that are relevant to the predicate of interest are the *relevant* events and messages, respectively. We assume that all events, variables, and messages recorded or logged are the relevant ones.

2 The Three-Phase Uncoordinated Snapshot Algorithm

The proposed algorithm is inspired by the two-phase deadlock detection algorithm [9]. The main idea of our algorithm is as follows. The algorithm takes three serial uncoordinated 'snapshots' that may be inconsistent. A consistent global state that lies between the first and the second inconsistent 'snapshots' is computed with the help of the third 'snapshot' and some local processing.

Any process can initiate the algorithm which consists of three phases that are serially executed. The algorithm involves some local processing by the initiator. Each phase involves the initiator sending a request to each other process, and then the processes replying to the initiator. The initiator can communicate directly to/from the various processes, or a *wave algorithm* [5] can be used in conjunction with a superimposed topology such as a ring or a tree. The snapshot algorithm is independent of this detail.

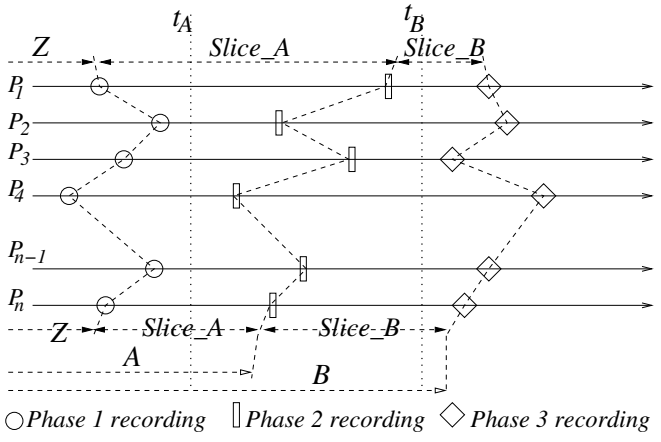


Fig. 1. Three-phase uncoordinated recording of a global snapshot. The initiator could be any of the processes.

Phase I: The initiator requests the processes to begin recording an execution slice. The global state from which processes begin recording is denoted Z, and is represented by the array $Z[1 \ldots n]$ at the initiator. The local states recorded in Z are not coordinated and may be inconsistent.

Phase II: The initiator then collects the slice of each process execution since the time each process began recording its slice and reported its local state in Phase I, until the time each process chooses to reply to the Phase II request. The local slice of each process that is reported to the initiator is stored by the initiator in array $Slice_A[1 \ldots n]$. Each process begins to record the next slice, denoted *Slice_B*, after replying to the Phase II request.

Phase III: The initiator then collects the slice of each process execution since the time each process reported its local state in Phase II, until the time each process chooses to reply to the Phase III request. The local slice of each process that is reported to the initiator is stored by the initiator in array $Slice_B[1 \ldots n]$. Based on *Slice_A* and *Slice_B*, the initiator computes a consistent global state.

Slice_B is used in two different ways, depending on whether channel states are to be recorded.

- If channel states are not to be recorded and the channels are non-FIFO, *Slice_B* is used to identify a consistent state within *Slice_A* by helping to eliminate states in *Slice_A* that are inconsistent.
- If channel states are also to be recorded and FIFO channels are assumed, then *Slice_B* is also useful to capture the channel states. In this case, the recording within *Slice_B* completes at each process when the messages sent by other processes to that process in (and before) *Slice_A* have been received. This condition is detectable using the control information distributively sent to the initiator in the Phase II reply messages and then conveyed on the Phase III request received from the initiator.

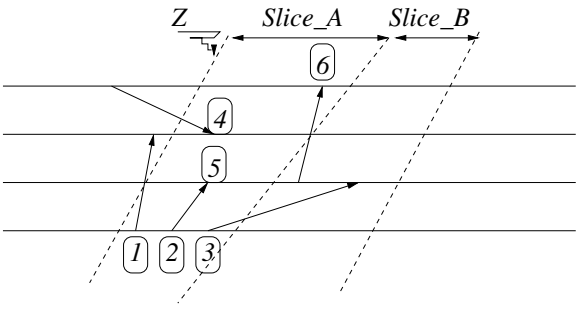


Fig. 2. Six types of events in *Slice_A*

The possibly inconsistent states collected by the initiator in Z, *Slice_A*, and *Slice_B* are illustrated in Figure 1. The initiator computes a consistent global state S, such that $Z \subseteq S \subseteq A$ using *Slice_A* and *Slice_B*. Specifically, observe that Z may be inconsistent because messages sent in *Slice_A* may have been received in Z. Also observe that due to the existence of global time instant t_A which is any time instant between the last recording of Phase I and the first recording of Phase II, no message sent in *Slice_B* could have been received before t_A. There exists at least one consistent cut in *Slice_A*, namely the cut at physical time t_A. However, as the application execution including its messages cannot be modified, and as timestamps are also not used in the algorithm, computing a consistent state S such that $Z \subseteq S \subseteq A$ is tricky. In *Slice_A*, there are six types of events (see Figure 2):

1. send event, for a message that gets delivered in Z
2. send event, for a message that gets delivered in *Slice_A*
3. send event, for a message that gets delivered after *Slice_A*
4. receive event, for a message that was sent in Z
5. receive event, for a message that was sent in *Slice_A*
6. receive event, for a message that was sent after *Slice_A*

To make Z consistent, we need to add that prefix from *Slice_A* that contains (i) all events of type (1) and no events of type (6), and (ii) the local states of processes are mutually consistent. Alternately, as A is also not consistent, A can be made consistent by subtracting that suffix that contains (i) all events of type (6), and (ii) further events to ensure that the resulting local states of processes are mutually consistent. With either approach, a consistent execution prefix exists, namely, the global state at t_A which satisfies both sets of conditions. Observe that all events of type (1) precede all events of type (6). Let $S(t)$ be the prefix of the execution at global time t. We now have the following.

– From Figure 1, we have $S(t_{Z_end}) \subseteq S(t_A) \subseteq S(t_{A_start})$, where t_{A_start} denotes the time instant of the first local recording of phase II among all the processes, and t_{Z_end} denotes the time instant of the last local recording of phase I among all the processes.

- Let S_{max} be Z + the largest prefix from *Slice_A* that does not include an event of type (6). Note that S_{max} may not be consistent.
- A tight lower bound on S_{max} is the value of $S(t_{A_start})$. A tight upper bound on S_{max} is the value of $S(t_{A_end})$, where t_{A_end} denotes the time instant of the last local recording of phase II among all the processes. Thus, $S(t_{A_start}) \subseteq S_{max} \subseteq A \subseteq S(t_{A_end})$.

The algorithm computes the largest consistent snapshot S such that $S(t_{A_start}) \subseteq S \subseteq S_{max} \subseteq A \subseteq S(t_{A_end})$, by removing the minimum slice suffix from S_{max} to get a consistent global state.

The third phase of recordings plays two roles.

- For both versions of the algorithm, *Slice_B* helps to identify S_{max} by identifying messages sent in *Slice_B* that were received in *Slice_A*.
- *Slice_B* also helps to identify the in-transit messages by ensuring that all the messages that have been sent up to and including in *Slice_A* are received before the recording of the end of *Slice_B*. This mechanism works only if FIFO channels are assumed.

The two versions of the algorithm are presented together in Figs. 3 and 4.

2.1 Consistent State Recording Under Non-FIFO Channels

This section describes a global snapshot algorithm that works with non-FIFO channels and satisfies properties (P1) to (P4). This algorithm records a consistent global state but does not capture channel states.

Figure 3 gives the code for the three-phase processing. The underlined pseudocode and data structures are ignored by this (version of the) algorithm. Step (1) describes the processing followed by the initiator. Step (2) describes the processing at all the nodes. In the first phase, an acknowledgement is sufficient from the nodes to the initiator; the local states are not required to be reported. When *Slice_A* and *Slice_B* are recorded, observe the following.

- Only the relevant local and send/receive events, of interest to the application or predicate being monitored, are recorded in the log of the slice.
- Messages are not modified with sequence numbers to conform to requirements (P1)-(P4). In addition, no counters for sequence numbers for the messages sent or received, or for the event count, are required at processes. A hash or checksum ($O(1)$ space) computed on each message sent or received is stored in the log of the slice, to enable matching a message in the sender's log with the same message in the receiver's log. No messages are stored.
- Within the log of a slice at a process, sequence numbers are assigned to events. However, these sequence numbers are of significance only to that process and within the slice. The sequence numbers have no global significance to the processes, or even within a process outside its slice. These numbers are used by the initiator to perform a simple ordering among the process events included in the slice.

(variables at an initiator)
array of states: $Z[1 \ldots n]$; // Phase 1 recordings
array of sequence of events: $Slice_A[1 \ldots n]$; // Phase 2 recordings
array of sequence of events: $Slice_B[1 \ldots n]$; // Phase 3 recordings
array of int: $Global_Sent[1 \ldots n, 1 \ldots n]$; //$Global_Sent[i, j] \equiv$ # msgs, P_i to P_j
array of int: $Global_Received[1 \ldots n, 1 \ldots n]$;
//$Global_Received[i, j] \equiv$ # messages received by P_i from P_j
(variables at each process)
array of local events: $Slice_Log$; // log of local events
array of int: $Sent[1 \ldots n]$; // $Sent[k] \equiv$ # messages sent to P_k
array of int: $Received[1 \ldots n]$; // $Received[k] \equiv$ # messages received from P_k
array of int: $Must_Receive[1 \ldots n]$;
// $Must_Receive[k] \equiv$ # messages to be recd. from P_k before Phase III report

(1) Process P_{init} initiates the algorithm, where $1 \leq init \leq n$.
(1a) **send** *Request(Phase_1_Report)* to all P_j;
(1b) **await** *Report(Phase_1_Report)* from all processes;
(1c) $(\forall j)$ $Z[j] \longleftarrow$ *Phase_1_Report.State* received from P_j;
(1d) $(\forall j)$ $Global_Received[j][1 \ldots n] \longleftarrow$ $Phase_1_Report.Received[1 \ldots n]$;
(1e) **send** *Request(Phase_2_Report)* to all P_j;
(1f) **await** *Report(Phase_2_Report)* from all processes;
(1g) $(\forall j)$ $Slice_A[j] \longleftarrow$ *Phase_2_Report.Slice_Log* received from P_j;
(1h) $(\forall j)$ $Global_Sent[j][1 \ldots n] \longleftarrow$ $Phase_2_Report.Sent[1 \ldots n]$ from P_j;
(1i) **send** to all P_j, *Request(Phase_3_Report)* + $Global_Sent[1 \ldots n][j]$ piggybacked;
(1j) **await** *Report(Phase_3_Report)* from all processes;
(1k) $(\forall j)$ $Slice_B[j] \longleftarrow$ Phase 3 report received from P_j;
(1l) $S \longleftarrow$ *Compute_Consistent_Snapshot*($Slice_A, Slice_B$);
(1m) *Compute_In-transit_Messages*(S).

(2) Process P_j executes the following, where $1 \leq j \leq n$.
(2a) On receiving *Request(Phase_1_Report)* from P_{init},
(2b) **send** local state and $Received[1 \ldots n]$ in *Report(Phase_1_Report)* to P_{init};
(2c) Begin recording log of events in $Slice_Log$;
(2d) On receiving *Request(Phase_2_Report)* from P_{init},
(2e) **send** $Slice_Log$ and $Sent[1 \ldots n]$ in *Report(Phase_2_Report)* to P_{init};
(2f) Reset $Slice_Log$;
(2g) On receiving *Request(Phase_3_Report)* and $Must_Receive[1 \ldots n]$ from P_{init},
(2h) Await until, $(\forall k), Received[k] \geq Must_Receive[k]$;
(2i) **send** $Slice_Log$ in *Report(Phase_3_Report)* to P_{init};
(2j) Stop recording events in $Slice_Log$.

Fig. 3. Three-phase algorithm to record a global snapshot. Underlined code is executed if channel states are needed in a FIFO system.

After $Slice_A$ and $Slice_B$ have been collected at the end of the three phases, the initiator invokes procedure *Compute_Consistent_Snapshot* in Figure 4 to compute a consistent cut S from $Slice_A$ and $Slice_B$. This cut S satisfies $S(t_{A_start}) \subseteq S \subseteq S_{max} \subseteq A \subseteq S(t_{A_end})$ and is computed by iteratively re-

(3) Process P_{init} **executes** *Compute_Consistent_Snapshot(Slice_A, Slice_B).*
array of int: S_{max}, S, T, U, V;
array of array of int: *Slice_A_events*$[1 \ldots i \ldots n][1 \ldots a_i]$;
array of array of int: *Slice_B_events*$[1 \ldots i \ldots n][1 \ldots b_i]$;
// Alternate representation of slices

(3a) **for** $i = 1$ **to** n **do**
(3b) **if** *Slice_A*$[i][x+1]$ is the 1^{st} receive in *Slice_A[i]* of a msg. sent in *Slice_B* **then**
(3c) $S_{max}[i] \longleftarrow x$
(3d) **else** $S_{max}[i] \longleftarrow a_i$;
(3e) $S, T, U \longleftarrow S_{max}$;
(3f) **for** $i = 1$ **to** n **do**
(3g) $V[i] \longleftarrow a_i$;
(3h) **repeat**
(3i) **for** $i = 1$ **to** n **do**
(3j) **for** $y = T[i] + 1$ **to** $V[i]$ **do**
(3k) **if** message(*Slice_A_events*$[i][y]$, *Slice_A_events*$[j][z]$) **then**
(3l) $U[j] \longleftarrow \min(U[j], z-1)$; // modify U to make it consistent
(3m) **if** $T = U (= S)$ **then return**(S);
(3n) $S \longleftarrow \min(T, U)$; // $S \equiv$ current upper bound on consistent state
(3o) $V \longleftarrow \max(T, U)$; //current upper bound on source of inconsistency
(3p) $T, U \longleftarrow S$;
(3q) **forever**.

(4) Process P_{init} **executes** *Compute_In-transit_Messages(S).*
(4a) $(\forall i \forall j)$ $transit(S[i], S[j]) \longleftarrow \emptyset$;
(4b) $(\forall i)$ compute *Global_Sent*$[i, 1 \ldots n]$ for $S[i]$ using *Slice_A*, *Global_Sent*$[i, 1 \ldots n]$;
(4c) **for** $j = 1$ **to** n **do**
(4d) **for** each successive event e_j^x in *Slice_A*$[j][1 \ldots a_j]$ and *Slice_B*$[j][1 \ldots b_j]$ **do**
(4e) **if** a message M was received from i (at this event with seq. # x) **then**
(4f) *Global_Received*$[j, i]++$;
(4g) **if** *Global_Sent*$[i, j] \geq$ *Global_Received*$[j, i]$ **and** $x > S[j]$ **then**
(4h) $transit(S[i], S[j]) \longleftarrow transit(S[i], S[j]) \cup \{M\}$.

Fig. 4. Finding a consistent state iteratively, and computing in-transit messages. Underlined code is executed if channel states are needed in a FIFO system.

moving the minimum slice suffix from S_{max} to get a consistent global state. For convenience, this procedure represents each of the two slices as an array $[1 \ldots n]$ of an array of integers. For example, *Slice_A_events*$[1 \ldots i \ldots n][1 \ldots a_i]$, where *Slice_A_events*$[i][j]$ denotes the jth event at process P_i, represents *Slice_A*.

In Figure 4, lines (3a)-(3d) identify S_{max}. Line (3e) initializes the integer vector variables S, T, and U to S_{max}. These vectors denote global states but the integer values denoting sequence numbers of the states in the slice have significance local to the initiator only. For example, $S[i]$ was assigned by P_i relative to the start of the slice and represents the sequence number of a local state of P_i in the slice. Lines (3f-3g) initialize the vector variable V to the state

at the end of *Slice_A*, namely, the state A. S is always set to the current known upper bound of the consistent global state that is sought. Vector V denotes the state that is the best known upper bound on the global state such that messages sent in the slice $V \setminus S$ may cause S to be inconsistent. Vectors T and U are working variables used to update S and V.

The main loop (3h)-(3q) updates S and V iteratively. In lines (3i)-(3l), U is used to track the prefix of the current S such that there are no inconsistencies caused by messages sent in $V \setminus S$. A message at the sender is matched with the same message at the receiver by comparing their hashes or checksums (line (3k)). If the message sent at $Slice_A_events[i][y]$ is received at $Slice_A_events[j][z]$, then $U[j]$ is updated to the minimum of its current value and $z - 1$ (line (3l)). An inconsistency, if any, is thus eliminated by removing the minimum suffix from the execution slice for the receiver P_j. However, messages sent in the slice $S \setminus U$ may still cause U to be inconsistent; this needs to be tested in the next iteration. Lines (3n)-(3p) initialize the values of S, T, U, and V for the next iteration. The procedure finishes when the loop (3i)-(3l) does not find any inconsistencies in the current value of S in line (3m).

2.2 Consistent State and Channel Recording Under FIFO Channels

This section presents an enhanced algorithm that also records channel states if channels are FIFO. Figure 3 gives the code for the three-phase processing. Underlined pseudo-code and data structures are also executed by this version of the algorithm. Procedure *Compute_Consistent_Snapshot* in Figure 4 computes a consistent cut from *Slice_A* and *Slice_B*, and is common to both versions of the algorithm.

Procedure *Compute_In_Transit_Messages* is used to compute the channel states. This procedure requires the data structures $Z[1 \ldots n]$ and integer arrays $Global_Sent[1 \ldots n, 1 \ldots n]$ and $Global_Received[1 \ldots n, 1 \ldots n]$ at the initiator during the processing of the algorithm. Integer vectors $Received[1 \ldots n]$, $Sent[1 \ldots n]$, and $Must_Receive[1 \ldots n]$ must also be maintained at each node. $Sent[j]$ and $Received[j]$ track the count of the number of messages sent to and received from process P_j, respectively. The main idea is simple. The state of channel $\langle P_i, P_j \rangle$ in a global state containing local states $S[i]$ and $S[j]$ at the processes, denoted as $transit(S[i], S[j])$, is simply those messages sent by P_i till state $S[i]$ that are not received until state $S[j]$ at P_j. One important difference from the previous version is that sequence numbers used to count events at a process are not local to a slice, but local to the entire execution of that process. This is to capture in-transit messages for channel states. Such messages could have been sent before *Slice_A* and need to be detected. To compute the channel state while satisfying conditions (P1) – (P4) and specifically that no sequence numbers can be tagged on messages, three issues need to be addressed.

1. Messages sent by P_i to P_j before state $S[i]$ must have reached P_j by the end of *Slice_B*.
   This is ensured by using the local *Sent* vector at each process and the *Global_Sent* array at the initiator. In the Phase II recording reported to

the initiator, the *Sent* vectors reported (line (2e)) are used to populate *Global_Sent* (line (1h)). The Phase III request sent to each process contains the piggybacked information about how many messages have been sent to that process by other processes (line (1i)). A process postpones its Phase III recording of the end of *Slice_B* until all these number of messages, remembered in array $Must_Receive$, have been delivered locally (line (2h)).

2. The set of messages sent by P_i to P_j up to the snapshot state $S[i]$, denoted here as $\mathcal{X}$, should be identifiable. There are two parts to this.
   - This set contains all the messages received by P_j from P_i with sequence numbers less than the value of $Sent[j]$ at $S[i]$. This value of $Sent[j]$ at $S[i]$ is computed (line (4b)) using *Global_Sent*, constructed from the Phase II report, and working backwards using the log *Slice_A*, also reported in Phase II. The resulting message count (i.e., the value of $Sent[j]$ at $S[i]$) is stored in-situ in the data structure $Global_Sent[i, j]$ as it is updated.
   - The messages received by P_j are enumerated as per the sequence numbers assigned by P_i, in lines (4d)-(4e). The enumeration of the sequence numbers is done in lines (4c)-(4f) using *Global_Received*, reported in Phase I to the initiator (line (1d)), and working forwards using the log *Slice_A* reported in Phase II and the log *Slice_B* reported in Phase III. The enumeration is done in-situ in $Global_Received[j, i]$.

   If $Global_Received[j, i] \leq Global_Sent[i, j]$ for a message, then that message belongs to $\mathcal{X}$.
3. The set of messages received by P_j from P_i after the snapshot state $S[j]$, denoted here as $\mathcal{Y}$, should be identifiable.
   These are the messages received from P_i at P_j in states numbered x such that $x > S[j]$.

From (2) and (3) above, $transit(S[i], S[j]) = \mathcal{X} \cap \mathcal{Y}$, is expressible as $\{M$ *recd by* P_j *at event* $x \mid Global_Received[j, i] \leq Global_Sent[i, j] \bigwedge x > S[j]\}$. Unlike the algorithm in Section 2.1, *Slice_Log*, *Slice_A*, *Slice_B* record messages for send and receive events if the contents of in-transit messages are required. The pseudo-code data structures do not reflect this for simplicity.

3 Complexity

The complexity analysis assumes a flat tree topology with the initiator as the root. A similar analysis can be conducted for the ring and more general tree topologies. Table 2 summarizes the complexity results. Note that the slices *Slice_A* and *Slice_B* are both *thin* slices, and their expected size is the same. Hence, the complexity is expressed in terms of *Slice_A* only. The expected width of *Slice_A* is the execution log that occurs in $\hat{rtt}_{max}$, the expected round-trip time between the two furthest nodes in the network. Let $max(owt_A)$ denote the maximum time for a message sent in *Slice_A* to reach its destination. The expected width of *Slice_B* is $max(rtt_{max}, owt_A)$ which is also $\hat{rtt}_{max}$.

Table 2. Complexity of the proposed non-inhibitory nonintrusive snapshot algorithm. Both *Slice_A* and *Slice_B* are *thin* slices, and their expected size is the same.

Metric	Recording snapshot (non--FIFO, no channel states)	Recording snapshot + channel states (w/ FIFO channels)
# messages	$6(n-1)$	$6(n-1)$
Msg. space (total)	$O(\|Slice_A\|)$	$O(\|Slice_A\|) + O(n^2)$
Time complexity (initiator)	$O(\|Slice_A\|)+$ $O(1/n \times \|Slice_A\|^2) + O(n^2)$	$O(\|Slice_A\|)+$ $O(1/n \times \|Slice_A\|^2) + O(n^2)$
Time complexity (non-initiator)	$O(\|Slice_A\|)$	$O(\|Slice_A\|) + O(n)$
Space complexity (initiator)	$O(\|Slice_A\|)$	$O(\|Slice_A\|) + O(n^2)$
Space complexity (noninitiator)	$O(1/n \cdot \|Slice_A\|)$ avg.	$O(1/n \cdot \|Slice_A\|)$ avg. $+O(n^2)$
Properties	No inhibition App. messages unmodified execution unmodified no log of history	No inhibition App. messages unmodified execution unmodified no log of history + *Sent*, *Received* vecs/process + *Global_Sent*, *Global_Receive* at init

4 Detecting Stable Predicates

The proposed algorithm to record a consistent global state can be used to detect any stable predicate. (See [7] for details.) Each process records the (possibly changing) values of the variables over which the predicate is defined, in *Slice_A*. When the initiator computes the consistent state S, it can also evaluate the

Table 3. Comparing algorithms to detect stable predicates

	Marzullo & Sabel [13]	Schiper & & Sandoz [15]	Proposed V.1	Proposed V.2
Detectable predicates	locally stable	strong stable	locally stable	all stable
overhead at nodes (= control msg overhead)	vector clock, $O(n)$ + entire log of msgs & events w/timestamps	vector clock, $O(n)$ + entire log of msgs & events w/timestamps	– event log in slice during 3-phase	*Sent/Received* $(O(n))$ + event & msgs. log in slice during 3-phase
App. msg overhead	vector clock, $O(n)$	vector clock, $O(n)$	–	–
processing	by initiator	by initiator	by initiator	by initiator
Channels	FIFO	non-FIFO	non-FIFO	FIFO
# control messages	$(n-1)$	$(n-1)$	$6(n-1)$	$6(n-1)$

predicate over these variables in state S. If the predicate is evaluated as true, then it is true and remains true henceforth because it is stable.

- Version 1 can detect locally stable predicates predicates.
- Version 2 can detect any stable predicate.

Table 3 compares the features and the complexities of the proposed algorithm with those of the algorithms by Marzullo-Sabel [13] and Schiper-Sandoz [15].

The full version of the results of this paper, including the correctness proofs and the complexity analysis, is in [7].

References

1. A. Acharya, B. R. Badrinath: Recording Distributed Snapshots Based on Causal Order of Message Delivery. Inf. Process. Lett. 44(6): 317-321 (1992)
2. S. Alagar, S. Venkatesan: An Optimal Algorithm for Distributed Snapshots with Causal Message Ordering. Inf. Process. Lett. 50(6): 311-316 (1994)
3. K. M. Chandy, L. Lamport: Distributed Snapshots: Determining Global States of Distributed Systems. ACM Trans. Comput. Syst. 3(1): 63-75 (1985)
4. C. Critchlow, K. Taylor: The Inhibition Spectrum and the Achievement of Causal Consistency. Distributed Computing 10(1): 11-27 (1996)
5. J.-M. Helary: Observing Global States of Asynchronous Distributed Applications. WDAG 1989: 124-135
6. G. S. Ho, C. V. Ramamoorthy: Protocols for Deadlock Detection in Distributed Database Systems. IEEE Trans. Software Eng. 8(6): 554-557 (1982)
7. A. Kshemkalyani, B. Wu: Detecting Arbitrary Stable Properties Using Efficient Nonintrusive Snapshots. Univ. of Illinois at Chicago, Tech. Report UIC-CS-03-05 (2005)
8. A. Kshemkalyani, M. Raynal, M. Singhal: An Introduction to Snapshot Algorithms in Distributed Computing. Distributed Systems Engineering 2(4): 224-233 (1995)
9. A. Kshemkalyani, M. Singhal: Correct Two-Phase and One-Phase Deadlock Detection Algorithms for Distributed Systems. IEEE SPDP 1990: 126-129
10. T.-H. Lai, T. Yang: On Distributed Snapshots. Inf. Process. Lett. 25(3): 153-158 (1987)
11. L. Lamport: Time, Clocks, and the Ordering of Events in a Distributed System. Commun. ACM 21(7): 558-565 (1978)
12. H. F. Li, T. Radhakrishnan, K. Venkatesh: Global State Detection in Non-FIFO Networks. IEEE ICDCS 1987: 364-370
13. K. Marzullo, L. S. Sabel: Efficient Detection of a Class of Stable Properties. Distributed Computing 8(2): 81-91 (1994)
14. F. Mattern: Efficient Algorithms for Distributed Snapshots and Global Virtual Time Approximation. J. Parallel Distrib. Comput. 18(4): 423-434 (1993)
15. A. Schiper, A. Sandoz: Strong Stable Properties in Distributed Systems. Distributed Computing 8(2): 93-103 (1994)
16. M. Spezialetti, P. Kearns: Efficient Distributed Snapshots. IEEE ICDCS 1986: 382-388
17. S. Venkatesan: Message-Optimal Incremental Snapshots. IEEE ICDCS 1989: 53-60

Load Balanced Allocation of Multiple Tasks in a Distributed Computing System

Biplab Kumer Sarker[1], Anil Kumar Tripathi[2], Deo Prakash Vidyarthi[3], Laurence Tianruo Yang[4], and Kuniaki Uehara[5]

[1] Faculty of Computer Science, University of New Brunswick, Fredericton, Canada
sarker@unb.ca
[2] Institute of Technology, Banaras Hindu University, Varanasi, India
anilkt@bhu.ac.in
[3] Jawaharal Nehru University, New Delhi, India
dpv@mail.jnu.ac.in
[4] Department of Computer Science, St. Francis Xavier University, Canada
lyang@stfx.ca
[5] Graduate School of Science and Technology, Kobe University, Japan
uehara@kobe-u.ac.jp

Abstract. A Distributed Computing Systems (DCS) calls for proper partitioning of tasks into modules and allocating them to various nodes so as to enable parallel execution of their modules by individual different processors of the system. A number of algorithms have been proposed for allocation of tasks in a Distributed Computing System. Most of the models proposed in literature consider modules of a single task for static allocation, for the purpose of allocation onto a DCS. Moreover, they did not consider the architectural capability of the processing nodes and the way of connectivity among them. This work considers allocation of disjoint multiple tasks with their corresponding modules and proposes a parallel algorithm for a realistic situation wherein multiple disjoint tasks with their modules compete for execution on an arbitrarily connected DCS based on well-known A* technique. The proposed algorithm also considers a load balanced allocation for the purpose. The paper justifies the effectiveness of the algorithm with the experimental results by comparing with previously reported works.

1 Introduction

In a distributed computing systems, processing nodes networked together, participate in various computational tasks to achieve minimum turn around time of the submitted tasks. The problem becomes more complex when the communicating modules in a task itself are assigned to different processing nodes to achieve the goal. Therefore, the inter module communication cost(IMC) needs to be minimized to obtain the minimum turn around time. On the other hand, the capacity of computational load for each processing node is needed to be considered for the whole system to maximize its throughput. This problem has been studied as task allocation or mapping problem in the literatures [2-6, 8-11]. Since

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 584–596, 2005.
© IFIP International Federation for Information Processing 2005

the problem is NP-hard [2, 9] and thus many heuristic solutions are possible for this problem.

Most of the algorithms for Task Allocation (TA) problem proposed by the scientists and researchers [2, 6] so far, do make one or more assumptions. These consider a single task partitioned into corresponding modules for the execution and the repercussion of a single task allocation on a DCS. Whereas in reality, a DCS receives number of tasks from time to time for the execution. Factually, a DCS facilitates concurrent execution of modules belonging to various unrelated tasks [7, 12, 13]. The modules of any particular task, having IMC, do cooperatively execute and do not depend on the modules of the other tasks. This leads to the situation wherein, a processing node may be assigned modules belonging to different tasks. It is to mention that the real issue of task allocation must not ignore the possibility of multiple modules assignment of various tasks to the processing nodes in a dynamic fashion [7].

Considering these view and furthermore, taking into account the architectural capability of the processing nodes and the optimality of the solution guaranteed by A* based TA [2], in this paper we present a parallel algorithm for load balanced allocation in a DCS.

The paper is organized as follows. The next section discusses the load parameter for multiple tasks which is used in our case as a cost function to minimize turnaround time. This is the basis of effectiveness of the allocation. In section 3, the A* algorithm for task allocation is proposed. Three examples are also illustrated in this section. In the next section, the comparative observations with the existing algorithms are presented. Section 5 also justifies the effectiveness and scalability of our algorithm. Finally, the work is concluded indicating the future directions.

1.1 Assumptions

As the task allocation problem remains to be NP-hard, various heuristic solutions are proposed with one or more assumptions [2]. This work also makes certain assumptions that are as follows.

1. The processing nodes in the DCS are heterogeneous. The tasks are disjoint and have no inter-task communication. Only the modules within a task have interdependencies and communication requirements.

2. Execution and communication matrices for the task graphs are assumed to be given. These matrices are different for every task and calculated in units of time. While partitioning the task into modules, we assume that the memory requirements of the modules are also calculated.

3. The assumption of the availability of interconnection graph accommodates nonregular type of interconnection networks.

Here, in this paper, the word ‘processor’ and ‘processing node’, ‘assignment’ and ‘allocation’ have been used to refer the same.

2 Load

The tasks submitted into a DCS are partitioned into suitable modules and then these modules are to be allocated to the processing nodes. Each task can be represented by a Task Graph $(TG) = (V_t, E_t)$, where (1) V_t is a set of vertices, each of which represents a module of the task $m_1, m_2, ..., m_n$ and (2) $E_t \subseteq V_t \times V_t$ is a set of edges each of which represents the Inter Module Communication (IMC) between the two modules at the end of the edge. We can also represent the network of processors $p_1, p_2, ..., p_n$ in a DCS as a Processor Graph $PG = (V_p, E_p)$; where vertices represent the processors and the edges represent the communication links between processors (see Fig.2). The goal of TA is to allocate the Task Graphs (TG) to a network of processors in a DCS (i.e. to PG) to achieve the minimum turn-around time of tasks [2].

A processor's load comprises of all the execution and communication costs associated with its assigned modules of the task [6]. The time required by the heaviest-loaded processor will determine the entire tasks' completion time. So, the TA problem must find a mapping of the set of m modules of l tasks to n processors so as to minimize tasks completion time. Our goal is to allocate the modules in such a way that does not cause any processing node to be overloaded because an overloaded node may affect adversely in the turn around time of the tasks in a heterogeneous DCS.

The load in a processing node p is calculated as follows

$$\sum_{l=1}^{k}\sum_{i=1}^{mi} X_{ilp}.M_{ilp} + \sum_{\substack{q=1 \\ q\neq p}}^{n}\sum_{l=1}^{k}\sum_{i=1}^{mi}\sum_{\substack{j=1 \\ j\neq i}}^{m}(C_{ijl} + CC_{pq}).M_{ilp}.M_{jlq} \qquad (1)$$

where, $CC_{pq} = C_{fi}.L^i_{pq}$

X_{ilp} = execution cost of i^{th} module of task l on processing node p
C_{ijl} = Inter-Module Communication(IMC)Cost between i^{th} and j^{th} module of task l
M_{ilp} = assignment matrix of i^{th} module of l^{th} task on processing node p

$$M_{ilp} = \begin{cases} 1 & \text{if module } m_i \text{ of task } l \text{ is assigned to processor } p \\ 0 & \text{otherwise} \end{cases}$$

M_{jlq} = assignment matrix of j^{th} module of l^{th} task on any other processing node q

$$M_{jlq} = \begin{cases} 1 & \text{if module } m_j \text{ of task } l \text{ is assigned to processor } q \\ 0 & \text{otherwise} \end{cases}$$

L^i_{pq}= connection matrix of two processors p and q, describing the links (direct/ single indirect/ double indirect etc.) of connection paths among the processing nodes in Processor Graph (PG).

C_{fi}= coefficient matrix which has n entries describing the IPC (Inter Processor Communication) costs for the links of connection paths among the processing nodes. For example, C_{f1}=5 (for direct connection between the processors), C_{f2}=10 (for processors which are indirectly connected by one link), C_{f3}= 20 (for processors which are indirectly connected by two links) etc.

The first part of the above equation 1 is the total execution cost of the modules of all the tasks allocated on a processing node p. The second part is the communication overhead on p with the modules of the tasks allocated on the other processing node such as q in the DCS. The i^{th} entry of the coefficient matrix C_{fi} corresponds to communication between two processors via i links. If processors p and q are not directly connected, we find L^2, multiply it by C_{f2}, (2^{nd} field of C_f), and check whether this comes out to be non-zero; if it does, we replace L^1 in calculation with L^2; if not, we find out $L^3$and multiply it with C_{f3} and check whether the product comes out to be non-zero. We continue like this until we find a non-zero value and then replace L^i in calculation with this (it is to be mentioned that we shall find a non-zero value within n multiplications, where n is the no. of processing nodes).

2.1 Global Table(GT)

To allocate the modules optimally so that no processor becomes overloaded, the load on each of the n processing nodes needs to be computed. By finding the processing node with heaviest load, the optimal assignment out of all possible assignments will allot the minimum load to the heaviest loaded processor. Thus it is necessary to consider realistic view that only a finite number of modules can be allocated to a processor depending on the architectural capability of the processing nodes in a DCS. Consequently, earlier algorithms [2, 5, 6] have continued to assume that all the modules will be eventually allocated no matter how large the memory requirements are, and/or how many modules a processor can accommodate and what is the current status of the system due to the existing allocation. These algorithms do not consider the requirement of allocation of modules of multiple tasks. In the proposed algorithm, we have shed off these unrealistic assumptions and make use of a data structure STATUS associated with every processor, which has two fields showing: the maximum number of modules that can be allocated to the processor and the memory capacity of the processor.

Whenever a module is chosen for allocation onto a processing node, the STATUS is checked and it is ascertained whether the processor can accommodate the module at hand. If not, another processor is chosen if available. The consequence might be that a certain module is not allocated at all. This data structure is implemented by constructing a Global table (GT) to maintain the track of maximum number of modules that can be allocated to a processing node depending upon its memory capacity. This is a dynamic table, which keeps the information of the remaining memory of nodes and the number of modules can be allocated on the nodes. Whenever a new task arrives, this GT is to be consulted and to

Table 1. The structure of the GT

P_{node}	M_{mod}	M_{cap}	Mod_{assign}	R_{mod}	R_{mem}
p_1	4	10			
p_2	3	8			
p_3	4	9			
p_4	5	12			

Here, P_{node}= Processing node of the DCS
M_{mod} =Maximum number of modules can be assigned
M_{cap} = Maximum memory capacity of a P_{node}
Mod_{assign} = Modules assigned of each task
R_{mod} = Remaining number of modules can be assigned
R_{mem} = Remaining available memory

be modified. Here, we present the basic structure of the GT. In this table, it is shown that there are 4 processing nodes in a DCS. The number of modules, the processing nodes can accommodate are 4, 3, 4, and 5, respectively, and the memory capacity of the nodes are 10, 8, 9 and 12 respectively. After some modules are assigned the other column of the processing nodes will be filled up by the corresponding numbers which is shown in the illustrative examples in sec. 4.

3 Proposed Algorithm for TA

In the A* algorithm [1, 2], for a tree search, it starts from the root, usually is called the start node (usually a null solution of the problem). Intermediate tree nodes represent the partial solutions, and leaf nodes represent the complete solution or goal. A cost function f computes each node's associated cost. The value of f for a node n, which is the estimated cost of the cheapest solution through n, is computed as

$$f(n) = g(n) + h(n) \tag{2}$$

Where, $g(n)$ is the search-path cost from the start node to the current node and $h(n)$ is a lower-bound estimate of the path cost from current node to the goal node (solution), using any heuristic information available. To expand a node means to generate all of its successors or children and to compute the f value for each of them. The nodes are ordered for search according to the cost; that is, the algorithm first selects the node with the minimum expansion cost. The algorithm maintains a sorted list, called OPEN, of nodes (according to their f values) and always selects a node with the best expansion cost. Because the algorithm always selects the best-cost node, it guarantees an optimal solution [2].

To compute the cost function, $g(n)$ is the cost of a partial assignment at node n which is the load on the heaviest loaded processing node (p_i); this is done using the equation 1. For the computation of $h(n)$, two sets A_p (the set of modules that are already assigned to the heaviest loaded p) and U (the set of modules that are

unassigned at this stage of the search and have one or more communication links with any module in set A_p), are defined. Each module m_i in U will be assigned either to p or any other processor q that has a direct or indirect communication link with p. So, two kinds of costs with each m_i's assignment can be associated: either X_{ilp}(the execution cost of m_i of task l on p) or the sum of communication costs of all the modules in set A_p that has a link with m_i. This implies that to consider m_i's assignment, it is to be decided whether m_i should go to p or not (by taking the minimum of these two cases' cost).

To support the run-time allocation of tasks to processors, we construct a manager-worker style parallel algorithm whose pseudo-code is given in sec. 3.1. One processor called the manager is responsible for keeping track of the assigned and unassigned tasks using a Global Table (GT) which is consulted and updated during every allocation. It always consists of the information about the total memory of the processing nodes and the remaining memory after assignment, no. of assigned modules and the remaining no. of modules can be assigned.

3.1 The Algorithm

1. *As a 'Manager' node, processor P_0 maintains the status of the Global Table (GT) for each processing node($P_1, P_2, ..., P_n$) termed as 'worker' in terms of available memory (M) and the modules that are already assigned to it.*
2. *'Manager' node maintains a list S of unallocated tasks with all modules (all tasks are in S at the beginning) and a list OPEN, empty at the beginning. Another list V is maintained by taking one Task ta from S and put it in another list V and reset OPEN.*
3. *The 'workers' checks possible allocation of modules in V using the A*(2) algorithm and verifying STATUS of them by P_0; then allocate them; if not possible, deallocate the partially allocated modules of the task and move onto the next task, modifying the STATUS in between and update the Global Table (GT)by the Manager.)*
4. *If S is not empty yet, go to step 2.*
5. *Stop (end of allocation).*

4 Implementation Results

In this section, we present three small examples with various number of TGs and PGs to justify the proposed algorithm with respect to allocation and status of the global table.

Case 1

For case 1, we have considered a set of three tasks shown as TGs partitioned with their corresponding modules $T_1(m_{11}, m_{21}, m_{31}, m_{41})$, $T_2(m_{12}, m_{22}, m_{32})$, $T_3(m_{13}, m_{23}, m_{33})$ and a DCS as PG, consists of four processors (p_1, p_2, p_3, p_4) interconnected as shown in Fig. 2. Here, the IMC costs shown as in the figure represent the communication costs between the modules of the tasks in time

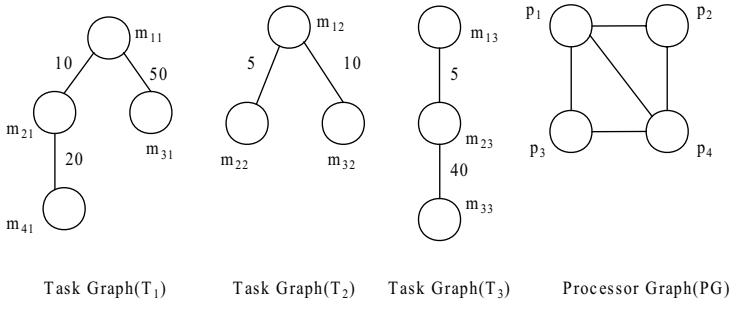


Fig. 1. Example of task graphs T_1, T_2 and T_3 with their modules and a DCS as processor graph

Table 2. The final status of the GT using A* for case 1

P_{node}	M_{mod}	M_{cap}	Mod_{assign}	R_{mod}	R_{mem}
p_1	4	10	$m_{21}m_{41}m_{22}$	1	1
p_2	3	8	$m_{12}m_{32}m_{23}$	0	2
p_3	4	9	$m_{13}m_{33}$	2	2
p_4	5	12	$m_{11}m_{31}$	3	5

unit. For example, the communication cost between m_{11} (the first module of task T_1) with m_{21} (the second module of task T_1) is 10 unit. The adjacency matrix L^i_{pq} of processing nodes are assumed to be given which represents how the processing nodes are connected among each other. For example, the processing nodes p_2 and p_3 are not directly connected, so $L^1_{p2p3} = 0$. But they are connected with at least one indirect link (through p_1 or p_4). So, $L^2_{p2p3} = 1$.

The Results for case 1

Total cost (communication and execution) at all the processing nodes is 500 units. Time required by the algorithm was 0.06 seconds. Note that the results are conducted in a single processor machine.

Case 2

The algorithm is implemented with other two cases. In case 2, a DCS consists of five tasks partitioned with their corresponding modules $T_1(m_{11}, m_{21}, m_{31}, m_{41}, m_{51}), T_2(m_{12}, m_{22}, m_{32}, m_{42}), T_3(m_{13}, m_{23}, m_{33}, m_{43}), T_4$ $(m_{14}, m_{24}, m_{34}, m_{44}, m_{54}, m_{64}, m_{74}), T_5(m_{15}, m_{25}, m_{35}, m_{45}, m_{55}, m_{65}, m_{75}, m_{85})$ and a set of five processing nodes $(p_1, p_2, p_3, p_4, p_5)$ interconnected in some fashion (Fig. 2).

The Results using case 2

Total cost at all the processing nodes is 1585 unit. Time required by the algorithm was 0.17 seconds.

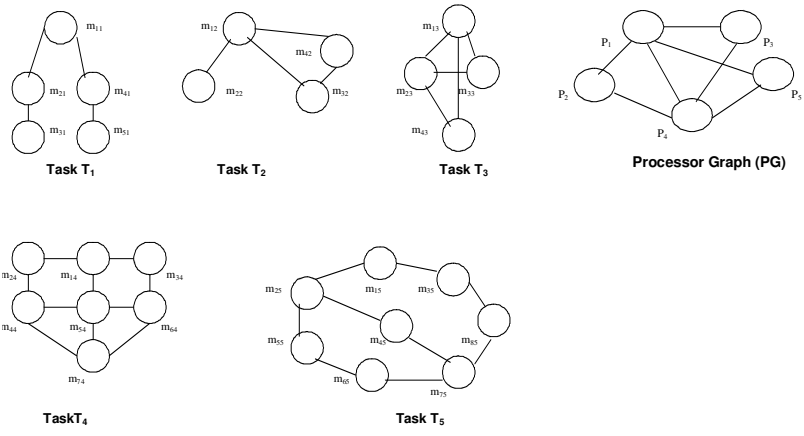


Fig. 2. Example of task graphs T_1, T_2, T_3, T_4 and T_5 with their modules and a DCS as processor graph

Table 3. The final status of the GT using A* for case 2

P_{node}	M_{mod}	M_{cap}	Mod_{assign}	R_{mod}	R_{mem}
p_1	10	50	$m_{21}m_{51}m_{12}m_{42}\ m_{33}m_{43}m_{14}m_{34}\ m_{64}m_{74}$	0	19
p_2	9	40	$m_{41}m_{22}m_{13}m_{24}\ m_{25}m_{85}$	3	21
p_3	7	35	$m_{32}m_{23}m_{44}\ m_{45}m_{65}$	2	21
p_4	6	30	$m_{11}m_{31}m_{54}m_{15}$	2	14
p_5	4	10	$m_{35}m_{55}m_{75}$	1	2

Case 3

A set of 8(eight) tasks with their corresponding modules $T_1(m_{11}, m_{21}, m_{31}, m_{41})$, $T_2(m_{12}, m_{22}, m_{32}, m_{42}, m_{52})$, $T_3(m_{13}, m_{23}, m_{33}, m_{43}, m_{53}, m_{63})$, $T_4(m_{14}, m_{24}, m_{34}, m_{44})$, $T_5(m_{15}, m_{25}, m_{35}, m_{45}, m_{55})$, $T_6(m_{16}, m_{26}, m_{36}, m_{46}, m_{56}, m_{66})$, $T_7(m_{17}, m_{27}, m_{37}, m_{47})$, $T_8(m_{18}, m_{28}, m_{38}, m_{48}, m_{58})$ and a set of 6(six) processors $(p_1, p_2, p_3, p_4, p_5, p_6)$ have been considered. Due to space limitation, here, we present only the final status of allocation using GT.

Table 4. The final status of the GT using A* for case 3

P_{node}	M_{mod}	M_{cap}	Mod_{assign}	R_{mod}	R_{mem}
p_1	10	50	$m_{21}m_{51}m_{12}m_{42}\ m_{33}m_{43}m_{14}m_{34}\ m_{64}m_{74}$	0	19
p_2	9	40	$m_{41}m_{22}m_{13}m_{24}\ m_{25}m_{85}$	3	21
p_3	7	35	$m_{32}m_{23}m_{44}m_{45}\ m_{65}$	2	21
p_4	6	30	$m_{11}m_{31}m_{54}m_{15}$	2	14
p_5	4	10	$m_{35}m_{55}m_{75}$	1	2

The Results using case 3

Total cost at all the processing nodes is 1380 unit. Time required by the algorithm was 0.19 seconds.

5 Comparative Observations

The TA algorithms that consider only modules of one task do not consider the limitation of memory and the number of modules that can be assigned to a particular processor. This is so because these algorithms are not meant for assignment of modules belonging to the multiple disjoint tasks. Such a single task assignment problem is easier to solve because of this reason.

Table 5. The final status of the GT using EA* for case 1

P_{node}	M_{mod}	M_{cap}	Mod_{assign}	R_{mod}	R_{mem}
p_1	4	10	$m_{21}m_{41}m_{22}m_{32}m_{23}m_{33}$	-2	1
p_2	3	8	$m_{12}m_{13}$	1	2
p_3	4	9		4	2
p_4	5	12	$m_{11}m_{31}$	3	5

However, we can execute the Single Task Allocation (STA) algorithms [2, 5, 6] multiple time ones for each task using the GT data structure to record the status of allocation and the system as done in our proposed algorithm (sec. 4). Now we compare the status of allocation and the execution time requirements of the method used in EA* and our proposed allocation algorithm. The STA based on A* [2] referred to as EA* in the subsequent discussion has been executed multiple times and the run times have been obtained. So, in the experiment, we have executed the tasks one by one for the cases 1, 2 and 3 without considering the processor connectivity (how the processor are connected i.e. with direct connection/indirect connection etc., because it is not possible to do so in EA*) for the EA* as described in the algorithm of [2]. In the work [6], another modified version of EA* is proposed but still it was developed for single task allocation with their modules by using the same idea of [2]. In [5], an algorithm has been presented to reduce the search space using the idea of [2, 6]. Therefore, here we

Table 6. The final status of the GT using EA* for case 2

P_{node}	M_{mod}	M_{cap}	Mod_{assign}	R_{mod}	R_{mem}
p_1	10	50	$m_{21}m_{51}m_{12}m_{42}\ m_{23}m_{43}m_{14}m_{34}\ m_{25}m_{85}m_{74}$	-1	19
p_2	9	40	$m_{41}m_{22}m_{13}\ m_{24}m_{55}$	4	21
p_3	7	35	$m_{32}m_{33}m_{44}\ m_{45}m_{65}$	2	21
p_4	6	30	$m_{11}m_{31}m_{54}$	2	14
p_5	4	10	$m_{15}m_{65}m_{75}$	1	2

Table 7. The final status of the GT using EA* for case 3

P_{node}	M_{mod}	M_{cap}	Mod_{assign}	R_{mod}	R_{mem}
p_1	10	70	$m_{11}m_{21}m_{41}m_{52}m_{13}m_{33}m_{63}m_{14}m_{34}m_{44}m_{15}$ $m_{55}m_{16}m_{46}m_{66}m_{17}m_{47}m_{18}m_{44}m_{18}m_{58}$	-10	35
p_2	8	50	$m_{31}m_{23}m_{24}m_{35}$ $m_{26}m_{27}m_{38}$	1	23
p_3	6	40	$m_{53}m_{36}m_{37}m_{38}$	2	20
p_4	7	35	$m_{43}m_{25}m_{45}m_{56}$	3	16
p_5	6	40	m_{32}	2	22
p_6	6	33	$m_{12}m_{22}m_{42}$	3	8

present the comparative results with the algorithm proposed in [2], as all the other STA algorithms are basically based on this.

If we look at the results shown in the Tables 5, 6 and 7 for allocation of tasks using EA* for the cases 1, 2 and 3, respectively, it is observed that balanced load allocation can not be achieved. In all the cases, presented in the tables, some processing nodes are overloaded (indicated by '-' sign) according to the V^{th} column of the GT considering their existing architectural capabilities. Thus, it is justified that the EA*, in the form, as reported in [2, 5, 6], can not be used for the allocation of multiple tasks. Here, A* is used to refer our A* based algorithm proposed in section 4. Furthermore, since we did not get good results using EA* in terms of allocation, hence we did not present here the running time and the total cost required by EA*.

5.1 Experimental Results

It is observed from results using A* in section 4 that the time taken by our algorithm is not very effective considering the small number of tasks and their corresponding modules. But the fact is that the experiments were conducted in a single machine to make comparisons with the earlier algorithms.

Therefore, to investigate the effectiveness and the scalability of our proposed algorithm we further experimented with large number of task graphs with corresponding modules. For simulation purpose we use Sun Fire 12K, 8 processors based Distributed Multiprocessor systems and Message Passing Interface (MPI) as programming environment. In the Fig. 3, 4 and 5, the x-axis represents the no. of tasks and y-axis represents the execution time in seconds. It is observed from the figures that for an amount of large number of tasks, our parallel algorithm performs better (Fig. 5) than the other two (Fig. 3 and 4) with respect to the execution time with an increasing number of processors. In Fig. 3, the 'pink' colored line represents the execution time for 60(approx.) tasks using 8 processing nodes and it is greater than the time required by all other processors. Fig. 4 represents the time required by our algorithm while number of tasks is equal to 100. Here also the 'pink' colored line indicates the number the running time taken by 8 processing nodes and it is still greater than the time required by the other processing nodes. Fig. 5 shows the time required while there is a

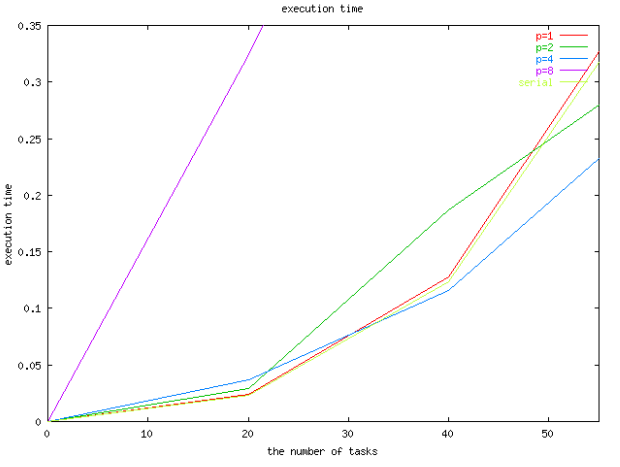


Fig. 3. Execution time using number of tasks = 60(approx.)

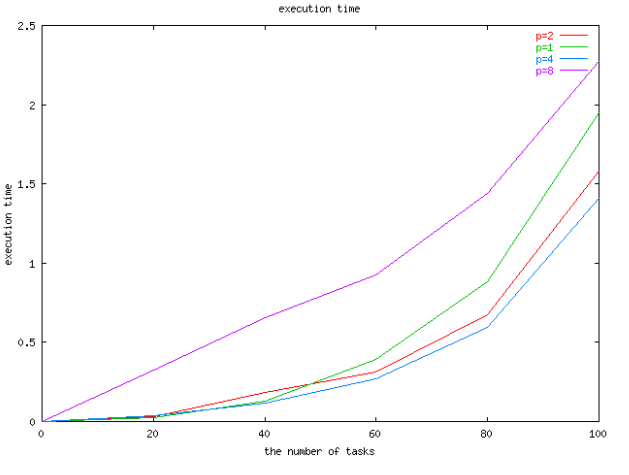


Fig. 4. Execution time using number of tasks = 100

large number of tasks is assigned i.e. the number of tasks is equal to 400 by the processing nodes. The results show that until the number of tasks is 200, the result is same using 8 processing nodes('pink' colored line) with the time required by other processors but as soon as the number of tasks increases(beyond 200), the timed required by 8 processors also decreases compared to the time required by other processors. Thus, we can say that for a large number of tasks our algorithm can perform well and scalable with the large number of increasing tasks. Note that the no. of tasks with their corresponding modules is generated randomly for these experiments.

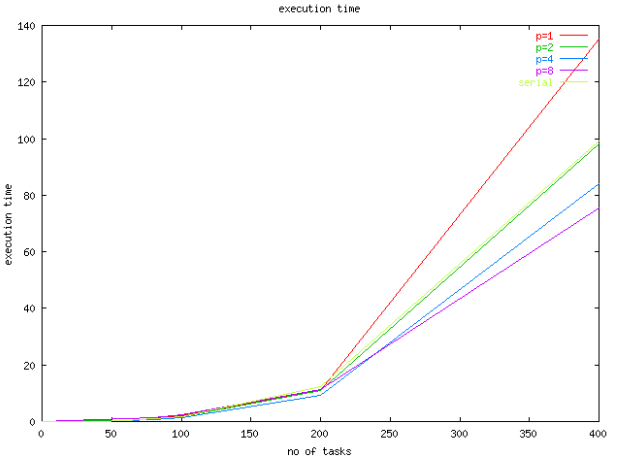


Fig. 5. Execution time using number of tasks = 400

6 Conclusion and Future Work

Our proposed algorithm has attributed the efficiency in allocating multiple tasks by optimizing a good load balanced among the processing nodes in a heterogeneous DCS. By realizing the dynamic situation of a system we have introduced a data structure Global Table and an algorithm which allocates the multiple tasks with their modules in such a way so that no processor becomes overloaded due to the allocation by considering their status based on their architectural capability.

Furthermore, we have conducted experiments for a large number of tasks with the corresponding modules. Comparing the results, obtained using our algorithm, it is evident that our proposed algorithm can provide effective solution in terms of scalability for the TA problem for a large number of tasks coming onto a DCS. As for future work we will concentrate on implementation of our proposed algorithm for a real time DCS.

References

1. N.J. Nilson, *Problem Solving Methods in Artificial Intelligence*. McGraw Hill International Edition, 1971.
2. C.C. Shen and W.H. Tsai, "A Graph Matching Approach to Optimal Task Assignment in Distributed Computing System Using A Minimax Criterion", *IEEE Transactions on Computers*, vol. C-34, no. 1, pp. 197-203, 1985.
3. A.K. Tripathi, D.P. Vidyarthi and A.N.Mantri, "A Genetic Task Allocation Algorithm for Distributed Computing System Incorporating Problem Specific Knowledge", *International Journal of High Speed Computing*, vol. 8, no. 4, pp. 363-370, 1996.

4. A.K. Tripathi, B.K. Sarker, N. Kumar and D.P. Vidyarthi, "A GA Based Multiple Task Allocation Considering Load", *International Journal of High Speed Computing*, vol. 11, no. 4, pp. 203-214, 2000.
5. M. Kafil and I. Ahmed , "Optimal Task Assignment in Heterogeneous Distributed Computing System", *IEEE Concurrency*, vol. 6, no. 3, pp. 42-51, 1998.
6. Ramakrishnan, H.Chao, and L.A.Dunning, "A Close Look at Task Assignment in Distributed Systems", *Proceedings of IEEE Infocom-91*, pp. 806-812, 1991.
7. D.P.Vidyarthi, A.K.Tripathi and B.K.Sarker, "Allocation Aspects in Distributed Computing System", *IETE Technical Review*, vol. 18, no. 6, pp. 279-285, 2001.
8. P.Y.R.Richard Ma, E.Y.S.Lee and J. Tsuchiya, "A Task Allocation Model for Distributed Computing Systems", *IEEE Transactions on Computers*, vol. C-31, no. 1, pp. 41-47, 1982.
9. S.H.Bokhari, "On the Mapping Problem", *IEEE Transactions on Computers*, vol. C-30, pp. 207-214, March, 1981.
10. Pradeep K. Sinha, Distributed Operating System, IEEE Press, Prentice Hall of India Ltd., 1998.
11. A.S.Tanenbaum, Distributed Operating Systems, Prentice-Hall, Englewood Cliffs, 1995.
12. A.K.Tripathi, B.K.Sarker, N.Kumar and D.P.Vidyarthi, "Multiple Task Allocation with Load Considerations", *International Journal of Information and Computing Science (IJICS)*, vol.3, no.1, pp. 36-44, 2000.
13. D.P.Vidyarthi, A.K.Tripathi and B.K.Sarker, "Multiple Task Management in Distributed Computing System", *Journal of the CSI*, vol. 31, no. 1, pp. 19-25, 2001.

ETRI-QM: Reward Oriented Query Model for Wireless Sensor Networks

Jie Yang, Lei Shu, Xiaoling Wu, Jinsung Cho, Sungyoung Lee, and Sangman Han

Department of Computer Engineering,
Kyung Hee University, Korea
{yangjie, sl8132, xiaoling, sylee, i30000}@oslab.khu.ac.kr
chojs@khu.ac.kr

Abstract. Since sensors have a limited power supply, energy-efficient processing of queries over the network is an important issue. As data filtering is an important approach to reduce energy consumption, interest is used to be a constraint to filter uninterested data when users query data from sensor networks. Within these interested data, some of them are more important because they may have more valuable information than that of the others. We use 'Reward' to denote the importance level of data. Among the interested data, we hope to query the most important data first. In this paper, we propose a novel query model ETRI-QM and a new algorithm ETRI-PF (packet filter) dynamically combines the four constraints: Energy, Time, Reward and Interest. Based on our simulation results, we find out that our ETRI-QM together with ETRI-PF algorithm can improve the quality of the information queried and also reduce the energy consumption. [1]

1 Introduction

Wireless sensor networks are envisioned to consist of large numbers of devices, each capable of some limited computation, communication and sensing, operating in an unattended mode. One unifying view is to treat them as distributed databases. The simplest mechanism to obtain information from this kind of database is to use queries for data within the network. However, most of these devices are battery operated, which highly constrains their life-span, and it is often not possible to replace the power source of thousands of sensors. So how to query with the limited energy resources on the nodes is a key challenge in these unattended networks.

Researchers have noted the benefits of a query processor-like interface to sensor networks and the need for sensitivity to limited power and computational resources [1, 3, 6, 7, 9,]. Prior systems, however, tend to view query processing in sensor networks simply as a power-constrained version of traditional query processing: given some set of data, they strive to process that data as energy-efficiently as possible. Typical strategies include minimizing expensive communication by applying aggregation and filtering operations inside the sensor network.

[1] Dr. Sungyoung Lee is the corresponding author.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 597 – 608, 2005.
© IFIP International Federation for Information Processing 2005

In our paper, we present a novel query model (ETRI-QM) to query the data with more important information among the interested data. By using this query model, we can dynamically combine these four constraints (Energy, Time, Reward, and Interest) to provide diverse query versions for different applications. Within our query model, each packet has four parameters: (1) energy consumption of the packet; (2) processing time of the packet; (3) important level of the packet; and (4) interest level of the packet. By using this ETRI-QM, we can achieve the following contributions: (1) Using *interest constraint* as the threshold to filter the uninterested incoming packets to reduce the energy consumption; (2) Using *reward constraint* to choose the high quality information and minimize the queried packet number to minimize the energy consumption but still satisfy the minimum information requirement.

The remainder of the paper is structured as follows. In the next section, we describe the related work. We introduce our query/event service APIs to illustrate the design of ETRI-QM in Section 3. The main principle of our ETRI-QM is in Section 4. In the simulation, we examine the performance of our query model and compare with the other different query plans (Section 5). Finally the paper is concluded in Section 6.

2 Related Work

In [8], the authors present a sensor information networking architecture called SINA, which facilitates querying, monitoring, and tasking of sensor networks. To support querying within sensor networks, they design a data structure kept inside the sensor nodes based on the spreadsheet paradigm. In the spreadsheet paradigm, each sensor node maintains a logical datasheet containing a set of cells. By defining the semantic of a cell to specifying scope of the query, the information can be organized and accessed according to specific application needs, and also the number of the packets need to be sent can be reduced, thus the energy consumption will be reduced. However, there exist a tradeoff between the energy cost to run SINA on each sensor node and the energy reduced by using SINA.

In [2], our work also has some similarities to techniques proposed, the authors introduced a new real-time communication architecture (RAP) and also a new packet scheduling policy called velocity monotonic scheduling (VMS). VMS assigns the priority of a packet based on its requested velocity. This work differs from our work in two aspects: one is that the cost-model is different in the two scenarios–in RAP is primarily reducing the end-to-end deadline miss radio while we are minimizing energy consumption and maximizing the querying quality; the second one is that RAP intends to maximize the number of packets meeting their end-to-end deadlines without considering their value (reward, importance level), and in our model, we take reward an important constraint to deal with the queries.

Samuel et al discussed the design of an acquisitional query processor (ACQP) for data collection in sensor network in [10]. They provide a query processor-like interface to sensor networks and use acquisitional techniques to reduce power consumption. Their query languages for ACQP focus on issues related to when and how often samples are acquired. To choose a query plan that will yield the lowest overall power consumption, the query is divided into three steps: creation of query, dissemination of query and execution of query. Optimizations are made at each step.

Our ETRI-QM combines four constraints (**energy, time, interest** and **reward**) to maximize the querying quality with minimum energy consumption. In papers [4, 5]. Cosmin Rusu, *et al.* first time consider **Energy**, **Time**, and **Reward** these three constraints simultaneously while Reward denotes the important level of tasks. They believe that among a set of tasks of real time applications, some of them are more valuable than the others. So instead of processing several unimportant tasks just consuming less energy, it is more meaningful to process one valuable task consuming more energy. In our query model, we use **reward** to denote the importance level of data, so that we can transmit the data with more valuable information first. By considering the four constraints simultaneously, we make out our target that is to query the most valuable (**reward**) packets from the **interested** area to be transmitted while meeting **time** and **energy** constraints.

3 ETRI-QM

Applications may submit queries or register for events through a set of query/event service APIs. The APIs provides a high-level abstraction to applications by hiding the specific location and status of each individual node. These APIs allow applications to specify the timing constraints as well as other constraints of queries.

ETRI-QM provides the following query/event service APIs.

Query {attribute_list, interested_area, system_value, timing_constraints, querier_loc}

Issue a query for a list of attributes in an interested area with the maximum system value (reward). Attributes refer to the data collected by different types of sensors, such as temperature sensors, humidity sensors, wind sensors, rain sensors etc. Interested area specifies the scope of the query, the area from which data is needed by the users. System value is defined as the sum of selected packets' reward. Timing_constraints can be period, deadline and so on. If a period is specified for a command, query results will be sent from the interested area to the issuer of query periodically. The querier_loc is the location of the base station that sends out the query.

Imagine a heterogeneous network consisting of many different types of sensors: temperature sensors, humidity sensors, wind sensors, rain sensors etc. monitoring the chemical found in the vicinity of a volcano. Suppose the volcano has just broken out, and we want to know which five chemicals found have the highest particle concentration. Obviously, sensors near the volcano will have more valuable data, which means that the importance levels of these data are much higher than those of the data collected by the further sensors. Thus, here we can consider reward to be the distance between sensors and the volcano.

Consider another example: lots of sensors are deployed in some area with different densities. For the EventFound case, take noise into account, the data collected by sensors having higher densities will be more reliable. So, here the reward is changed to be the density of the sensors around the interested area in the network.

There is one more example to make you clearly understand the concept of "reward". In the case of real-time communication for wireless sensor network, meeting the end-to-end deadline seems to be the most importance issue, so that we can consider arriving time to be the reward value. Reward is defined to be the importance

level of the data collected by sensors. In the case of different wireless sensor networks, it can be specified to various formats.

A query is send to every node in the interested area specified in the API, and the results will first be sent back to the cluster head, then the cluster head will use the algorithm we will introduce in next section to decide the packets to be sent back to the base station of which the location is also provided by the API.

4 ETRI-PF

After receiving the query message, the sensor nodes will start to collect related data and then send the packets to the cluster head. The cluster can be formed using LEACH or other techniques. In terms of the cluster head, many unprocessed packets are still physically existing in different sensor nodes and waiting for the processing of cluster head. Therefore, in sensor network, except the cluster head, all the other sensor nodes which are going to send packets to the cluster head can logically be considered as a buffer, since all of these packets are waiting for the processing of cluster head. We regard this buffer as the **First Tier Buffer (FTB)**. Actually the FTB is a logical concept for cluster head. The **Second Tier Buffer** (**STB**) is the buffer that physically exists inside cluster head. Since many sensor nodes will send packets to cluster head, obviously, cluster head needs buffer to store these received packets. Therefore, we propose the Two Tiers Buffer model for wireless sensor network as the figure 1 shows.

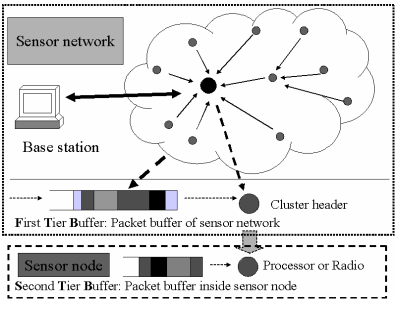


Fig. 1. Two tiers buffer

The main contribution of this paper is the ETRI-PF algorithm used in FTB to filter and accept packets. In FTB, what we want is to **Maximize reward value to meet the Reward constraint (in terms of system_value in the query/event service APIs)**. The key idea of this algorithm is that instead of processing two or more unimportant packets which just consume a small amount of energy we would like to process one important packet which may consume relatively larger amount of energy. Reward value is used to denote the important level of packet. A packet with a larger reward value means that this packet is more important. Therefore, the sensor nodes always

accept packets which have the highest reward value. Thus, we can guarantee that the most important packets can be processed first.

After deciding which packets are to be accepted, the algorithm will also arrange the packets according to their value. Packets with the largest value will be sent to STB first, meanwhile, FTB will sum the reward value of all the packets having been sent to STB. If the summation is up to the system_value defined in the query/event service APIs, no more packets will be sent to STB. That is to say, all the packets having been sent to STB is enough to solve the query.

Based on this Two Tiers Buffer model and the algorithms above, we introduce the details of our ETRI packet scheduling principles. The principles of ETRI-PF are as follows:

(1) Whenever a new packet is accepted, its energy consumption should not exceed the remaining energy;
(2) Whenever a packet is processed, it must meet its deadline;
(3) Every packet can under Energy, Timing, Reward, and Interest constraints simultaneously;
(4) It is not necessary to always under these four constraints at the same time;

We can dynamically compose these constraints to filter and schedule packet for heterogeneous sensor nodes and divers working purposes.

4.1 Problem Formulation

We define the interested areas as $A \subseteq \{A_1, A_2 \ldots A_M\}$. From each interested area A_x the cluster head can accept a subset of packets $P_x \subseteq \{ P_{x,1}, P_{x,2}, \ldots, P_{x,N} \}$.The processing time of the packet $P_{x,y}$ is denoted by $T_{x,y}$. Associated with each packet $P_{x,y}$ there is an Interest value $I_{x,y}$ and a Reward value $R_{x,y}$. Interest value is used to distinguish the interested packets from different areas. Reward value is used to denote the important level of this packet. The larger reward value means the higher important level. These four constraints of algorithm are defined as follows:

- The *energy constraint* imposed by the total energy E_{max} available in the cluster head. The total energy consumed by the accepted packet should not exceed the available energy E_{max}. In other words, whenever the cluster head accept one packet, the energy consumption $E_{x,y}$ of this packet should not be larger than the remaining energy RE.
- The *time constraint* imposed by the global deadline D. The common deadline of this user's data query is D. Each packet that is accepted and processed must finish before D.
- The *interest constraint* imposed by the interest value threshold IT. Each packet that is accepted and processed must satisfy the interest value threshold $IT_{min} \leq I_{x,y} \leq IT_{max}$.
- The *reward constraint* imposed by the *value ratio* $V_{x,y}$ ($V_{x,y} = R_{x,y} / E_{x,y}$) between reward value $R_{x,y}$ and energy consumption of packet $E_{x,y}$. The larger $V_{x,y}$, the packet has, the more valuable the packet is.

The ultimate goal of ETRI-PF is to query a set of packets $P = P_1 \cup P_2 \cup \ldots \cup P_M$ among interested packets to maximize the *system value* which is defined as the sum of

selected packets' *value ratio* $V_{x,y}$ to meet the system_value defined in the query/event service APIs. Therefore, the problem is to

$$\text{Maximize} \quad \sum_{x\in A, y\in P} V_{x,y} \quad \leq \text{system_value} \tag{1}$$

$$\text{Subject to} \sum_{x\in A, y\in P} E_{x,y} \leq E_{\max} \tag{2}$$

$$\sum_{y\in P} T_{x,y} \leq D \tag{3}$$

$$IT_{\min} \leq I_{x,y} \leq IT_{\max} \tag{4}$$

$$x \in \mathrm{A} \tag{5}$$

$$\mathrm{A} \subseteq \{A_1, A_2 \ldots A_M\} \tag{6}$$

$$y \in P_x \tag{7}$$

$$P_x \subseteq \{1, 2 \ldots \mathrm{N}\} \tag{8}$$

Since $\mathrm{P} = P_1 \cup P_2 \cup \ldots \cup P_M$, we can have the following equation as:

$$\sum_{x\in A, y\in P} V_{x,y} = \sum_{A_1, y\in P_1} V_{A_1,y} + \sum_{A_2, y\in P_2} V_{A_2,y} + \ldots + \sum_{A_M, y\in P_M} V_{A_M,y} \tag{9}$$

From equation (9), we can find that the real problem of ETRI-PF is to find out the minimum subset of $P_x \subseteq \{1, 2 \ldots \mathrm{N}\}$ to maximize the *system value* to system_value from each interested area A_x. Thus, the problem is changed to

$$\text{Maximize} \quad \sum_{A_x, y\in P_x} V_{x,y} \quad \leq \text{system_value} \tag{10}$$

$$\text{Subject to} \quad E_{x,y} \leq RE \tag{11}$$

$$\sum_{y\in P} T_{x,y} \leq D \tag{12}$$

$$IT_{\min} \leq I_{x,y} \leq IT_{\max} \tag{13}$$

$$x \in \mathrm{A} \tag{14}$$

$$\mathrm{A} \subseteq \{A_1, A_2 \ldots A_M\} \tag{15}$$

$$y \in P_x \tag{16}$$

$$P_x \subseteq \{1, 2 \ldots \mathrm{N}\} \tag{17}$$

Inequality (11) guarantees that the *time constraint* is satisfied. Inequality (12) guarantees that only the interested packets are accepted, and inequality (13) guarantees that the energy budget is not exceeded. In order to solve the problem that is presented by (10)-(17), we give the following steps for our ETRI-PF algorithm.

4.2 Steps of ETRI-PF

Before sending the real data of a packet to cluster head, sensor node can send its packet's parameters to the cluster head by including them in a small packet, which just consumes very limited energy. We give a name to this kind of small packet as *Parameter Packet (PP)*. There is a physical buffer that exists inside cluster head to

store these *PPs*. After receiving these *parameter packets*, cluster head can decide which packet to be accepted and which packet should be discarded based on these sent parameters. In terms of this Two Tiers Buffer model, basically, we can define our ETRI-PF algorithm into the following steps:

Step 1: Initialization. After receiving PP $\subseteq \{PP_1, PP_2, \ldots, PP_N\}$, we assume that tables exist inside the cluster head for storing parameters of every packet i ($i \in$ PP): *energy consumption* $E_{x,y}$, *processing time* $T_{x,y}$, *reward value* $R_{x,y}$, and *interest value* $I_{x,y}$. For each PP_i, there are energy consumption for checking CE_i and a period of time for checking CT_i. We also use two structure arrays, *considered(i)* and *selected(i)* of size N, to store the information for all received *PPs*. Initially, we start with an empty schedule (*selected(i).status = false*) and no *PP* is considered (*considered(i).status = false*). The set of selected *PPs* (initially empty) is defined as S = {*(i)* | *selected(i).status = true*}. After selecting the *PPs*, cluster head accepts packets that are corresponded to these selected *PPs*. Therefore, packet's parameters can be expressed as *considered(i).*$E_{x,y}$, *considered(i).*$T_{x,y}$, *considered(i).*$R_{x,y}$, *considered(i).*$I_{x,y}$, *selected(i).*$E_{x,y}$, *selected(i).*$T_{x,y}$, *selected(i).*$R_{x,y}$, and *selected(i).*$I_{x,y}$. We define five variables: 1) *checking energy* ($\sum_{i \in \text{PP}} CE_i$) is used to store the total energy consumption for checked *PPs*; 2) *checking time* ($\sum_{i \in \text{PP}} CT_i$) is used to store the total processing time for checked *PPs*; 3) *processing energy* ($\sum_{i \in \text{PP}}$ *selected(i).*$E_{x,y}$) is used to store the total energy consumption for processed packets; and 4) *processing time* ($\sum_{i \in \text{PP}}$ *selected(i).*$T_{x,y}$) is used to store the total processing time for processed packets. 5) *system value summation* ($\sum_{i \in \text{PP}}$ *selected(i).*$R_{x,y}$) is used to store the total value for packets to be processed in STB. These five variables are all initialized to zero.

Step 2: In FTB, we filter and accept packets based on the ETRI constraints. A packet that can be accepted should satisfy all the following criteria:

- This packet's *PP* is not considered before (*considered (i).status = false*).
- The current schedule is feasible (*checking time + processing time*) $\leq D$.
- By accepting this packet to current schedule, the energy budget is not exceeded (*checking energy + processing energy + considered(i).*$E_{x,y} \leq E_{max}$).
- This packet is intentionally queried by end user ($IT_{min} \leq$ *considered(i).*$I_{x,y} \leq IT_{max}$).
- Among all the *PPs* that satisfy the above criteria, select the one that has the largest *considered(i).*$V_{x,y}$ = *considered(i).*$R_{x,y}$ / *considered(i).*$E_{x,y}$.
- By accepting this packet to current schedule, the summation of the system value is just up to the system_value defined in the query/event service APIs. ($\sum_{i \in \text{PP}}$ *selected(i).*$V_{x,y} \leq$ *system_value*)

After choosing the *PP*, cluster head can send Acknowledge back to accept new packet. In addition, for those packets which end user is not interested in, their corresponded sensor nodes will discard them. In this case, we refuse and discard the unnecessary data; consequently, we can reduce the energy consumption by reducing the data transmitting and receiving.

Step 3: In STB, we transmit accepted packets to base station by using Velocity Monotonic Scheduling:

As the algorithm that has been presented in [2], which assigns the priority of a packet based on its requested velocity. VMS minimizes the deadline miss ratios of

sensor networks by giving higher priority to packets with higher requested velocities, which also reflects the local urgency. VMS embodies with both the timing constraint and location constraint.

Another aspect: Replace or drop a packet in the STB. A new packet is always accepted if possible. When receiving new *PP* from sensor node, if the STB is full, we can replace or drop a packet based on the following criteria:

- This packet's *PP* is selected (*selected(i).status = true*).
- Among all selected packet's *PPs*, find out the one that has the smallest *selected(i).*$V_{x,y}$ *= selected(i).*$R_{x,y}$ */ selected(i).*$E_{x,y}$.
- If this found one is not the new packet that is going to be accepted, we use this new packet to replace this found one, otherwise, we drop this new packet.

The flowchart and source code of ETRI-PF principles are showed in figure 2 and 3.

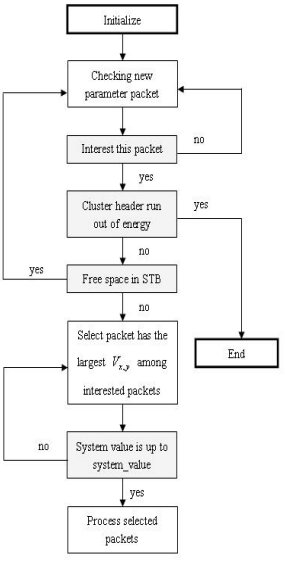


Fig. 2. Flowchart of ETRI-PF

```
Step1  initialize: selected[i].status = false; considered[i].status = false;
       ∀i ∈{1,2,…,N}; checking_energy = 0; checking_time = 0;
       processing_energy = 0; processing_time = 0;
Step2  if(considered[i].interest_value >= interest_threshold) then
           considered[i].status = true;
           if(cluster header buff has free space) then
               receive_pp();
           else
               if(current_pp.ratio > min(pp.ratio in buff)) then
                   receive current_pp; discard min(pp.ratio in buff);
               else
                   discard current_pp;
               end if
           end if
       end if
       calculate_checking_time();
       calculate_checking_energy();
       for i to buff_size
           if(checking_time + processing_time > deadline) then
               stop_working();
           else
               if(total_ratio < system_value) then
                   if(checking_energy + processing_energy <= total_energy)
                       selected[i].status = true;
                   else
                       change selected[i] with
                       max(selected[i].processing_energy in select);
                   end if
               end if
           end if
       next
Step3  for i to buff_size
           if(selected[i].status = true) then
               receive_packet(); process_packet();
               calculate_processing_time(); calculate_processing_energy();
           end if
       Next
Step4  End
```

Fig. 3. Pseudo code of ETRI-PF

5 Simulation Results

A sensor network can be modeled as a graph, where each vertex represents a sensor node and each edge represents the edge between two nodes when they are within each other's communication range. This network tracks the values of certain variables like temperature, humidity, etc. Application users submit their requests as queries and the sensor network transmits the requested data to the application.

For the simulation work, we randomly deploy eleven different sensor nodes. And we randomly initialize these sensor nodes with: the total energy of sensor nodes (scope: from 111 to 888), the buffer size of sensor nodes (scope: from 6 to 9). Ten of these eleven sensor nodes are chosen to be the packet generators which randomly create these

ten different packets and send to the remaining one. The remaining one works as the cluster head. For this cluster head, we design five parameters: the *total energy = 666*, the *buffer size = 6*, the *deadline = 5*, the *system_value = 10* and the *interest threshold = 5*. The meaning of threshold is that we just accept the packets when their interest value are larger than 5. Packets from those areas are what the end users are interested in.

In addition, we design ten different packets that are randomly initialized with the following four parameters: energy consumption (scope: from 3 to 10), processing time (scope: from 3 to 10), reward value (scope: from 3 to 10) and interest value (scope: from 3 to 10).

These ten sensor nodes are organized into three groups based on their created packets' interest values. The packets that have the interest values belong to {8, 9, 10} are considered as group A, the packets that have the interest values belong to {6, 7} are considered as group B, and the packet that have interest values belong to {3, 4, 5} are considered as group C. Suppose the cluster head just accepts the packets from area A and B, moreover, within these interested packets it accepts the packet that has the largest $V_{x,y} = R_{x,y} / E_{x,y}$ first. And we also design that this cluster head works in the STB by using the *Velocity Monotonic Scheduling*.

In terms of energy consumption, we mainly consider the following two parts that have strong relationship with our proposed ETRI-PF, which are *processing energy* $\{E_{(Returning\ ACK)} + E_{(Receiving\ packet)} + E_{(Processing)} + E_{(Broadcasting\ event)} + E_{(Listening)} + E_{(Accepting\ ACK)} + E_{(Sending\ packet)}\}$ and *checking energy* $\{E_{(Accepting\ event)} + E_{(Deciding)}\}$. The checking energy is designed to be 0.3, which is 10% of the minimum packet consumption 3; also the checking time is designed to be 0.3, which is 10% of the minimum processing time 3. Besides ETRI-PF, we provide two different existing packet scheduling algorithms to run on the cluster head for comparison as follows:

1) Compared Algorithm one (CA 1):
   a) In FTB: No *interest constraint* and no *reward constraint*
   b) In STB: Minimizing the packet deadline miss ratio (*Velocity Monotonic Scheduling*)

The cluster head doesn't set any threshold to reduce the incoming packets, but just simply receives packets and relays them. Once it gets a packet, it will process this packet based on the *Velocity* determined by *time constraint* and *location constraint*.

2) Compared Algorithm two (CA 2):
   a) In FTB: Consider *interest constraint*, but no *reward constraint*
   b) In STB: Minimizing the packet deadline miss radio (*Velocity Monotonic Scheduling*)

The cluster head always accepts the packet that has the interest value larger than the interest threshold. Once it gets a packet, it will process this packet based on the *Velocity* determined by *time constraint* and *location constraint*.

We used the following metrics to capture the performance of our routing approach and to compare it with other algorithms:

1) *total processing energy* of cluster head, 2) *energy utilization* of cluster head (*energy utilization = processing energy / (checking energy + processing energy)*), 3) *discarded packets ratio* in sensor nodes (*discarded packets number / total created packets number* by sensor nodes), 4) *total time consumption* of cluster head (*checking time + processing time*), 5) *average interest value* per packet, 6) *average reward value* per packet. The simulation results and comparisons are showed as the following figures.

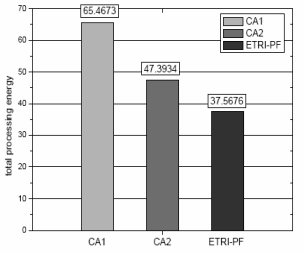


Fig. 4. Total processing energy

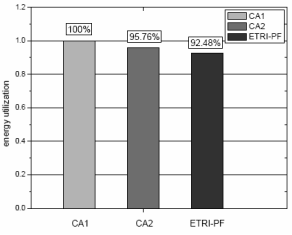


Fig. 5. Energy utilization

From figure 4, we can find that algorithm CA1 costs a lot of *processing energy* and our ETRI-PF algorithm costs only about half of that. The reason is that the cluster header just simply receives the packets and relays them without reducing any incoming packets, neither interest nor reward constraint is considered in algorithm CA1. Take a look at figure 5, we find that the *energy utilization* (= *processing energy / (checking energy + processing energy)*) of our ETRI-PF algorithm is a little bit lower than the other two algorithms. Remember that we used both interest and reward constraints, which would definitely cost some checking energy, however, we still reduce the energy consumption of whole sensor networks. The saved energy comes from the normal sensor nodes but not from the cluster head.

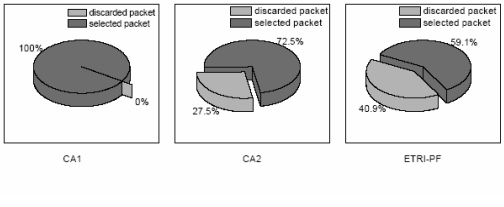


Fig. 6. Discarded packet ratio

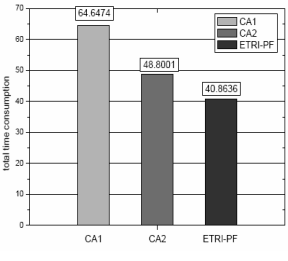


Fig. 7. Total time consumption

Same conclusion can also be drawn form figure 6, by analyzing the *discarding ratio* (*discarding ratio = discarded packets / total created packets*), we can see that the discarding ratio of our ETRI-PF is much higher than others. The lower *discarding ratio* the sensor nodes have, the more uninterested packets the sensor nodes send. Thus, the more unnecessary energy is consumed. In conclusion, by using the ETRI-PF, the sensor nodes can reduce the unnecessary transmission of uninterested data to reduce the energy consumption.

Consequently we get figure 7 showing the *total time consumption*, even though we need more checking time, we reduce the total time consumption by processing only part of the packets. For this part, the packets have larger reward than that of the rest packets.

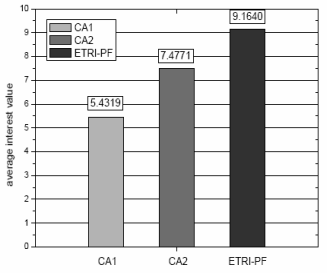


Fig. 8. Average interest value

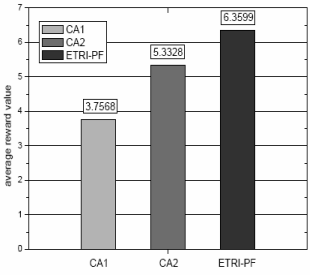


Fig. 9. Average reward value

As we presented in foregoing paragraph, we design the *interest threshold* to accept packets that have larger interest values, therefore, the desired *average interest value* should be larger than that of other algorithms. Figure 8 shows that the *average interest value* of ETRI-PF is much larger than others, which means the ETRI-PF can exactly process the interested packets well. Figure 9 shows the comparison among three algorithms' *average reward values*. In the algorithm CA 1, because we do not intentionally maximize the *value ratio* ($V_{x,y} = R_{x,y} / E_{x,y}$), as a result, the *average reward value* of CA 1 is relatively smaller than others. Compared with CA 2, even though we add the *interest constraint* to CA 2, still no reward constraint is considered, thus the *average reward value*s of our ETRI-PF is the largest one. Once again, we demonstrate that our ETRI-QM can deal with the queries more efficiently and get more important information to solve the queries.

6 Conclusion and Future Work

Wireless sensor networks consist of nodes with the ability to measure, store, and process data, as well as to communicate wirelessly with nodes located in their wireless range. Users can issue queries over the network. Since the sensors have typically only a limited power supply, energy-efficient processing of the queries over the network is an important issue. In this paper, we proposed a novel query model ETRI-QM dynamically combining the four constraints: Energy, Time, Reward and Interest. By considering these four constraints simultaneously, we can maximize the system value among the interested packet while satisfying the time and energy constraints by using our ETRI-PF algorithm. In this algorithm, we choose to process packets which have the highest reward value. A packet with a larger reward value means that this packet is more important. Based on our simulation results, we find out that our ETRI-QM and ETRI-PF algorithm can improve the quality of the information queried and also reduce the energy consumption.

However, as we mention the ETRI-QM principle that sensor nodes can know the reward value and interest value of packets well. In the simulation we randomly design the interest value and reward value for 10 different packets. But we do not mention the method that how to design the reward value and interest value for different

packets based on each packet's content. Therefore, as a challenge issue to be solved in the future, we are going to explore the appropriate measure methods to evaluate the interest level and important level of different packets.

Acknowledgement

This work was supported by grant No. R01-2005-000-10267-0 from Korea Science and Engineering Foundation in Ministry of Science and Technology.

References

1. C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *MobiCOM*, Boston, MA, August 2000.
2. C. Lu, B. M. Blum, T. F. Abdelzaher, J. A. Stankovic, and T. He, RAP: A Real-Time Communication Architecture for Large-Scale Wireless Sensor Networks, In IEEE RTAS 2002.
3. S. Madden and M. J. Franklin. Fjording the stream: An architechture for queries over streaming sensor data. In *ICDE*, 2002.
4. C. Rusu, R. Melhem, D. Mosse, "Maximizing Rewards for Real-Time Applications with Energy Constraints", ACM TECS, vol 2, no 4, 2003.
5. C. Rusu, R. Melhem, D. Mosse, "Multi-version Scheduling in Rechargeable Energy-aware Real-time Systems", to appear in Journal of Embedded Computing, 2004.
6. S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: A Tiny AGgregation Service for Ad-Hoc Sensor Networks. In *OSDI (to appear)*, 2002.
7. P.Bonnet, J.Gehrke, and P.Seshadri. Towards sensor database systems. In *Conference on Mobile Data Management*, January 2001.
8. Chien-Chung Shen, Chavalit Srisathapornphat, and Chaiporn Jaikaeo. Sensor information networking architecture and applications. IEEE Personal Communications, 8(4):52-59, August 2001.
9. O. Wolfson, A. P. Sistla, B. Xu, J. Zhou, and S. Chamberlain. DOMINO: Databases fOr MovINg Objects tracking. In *ACM SIGMOD*, Philadelphia, PA, June 1999.
10. Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. The Design of an Acquisitional Query Processor for Sensor Networks. SIGMOD, June 2003, San Diego, CA

Performance of Signal Loss Maps for Wireless Ad Hoc Networks

Henry Larkin, Zheng da Wu, and Warren Toomey

Faculty of IT, Bond University, Australia
{hlarkin, zwu, wtoomey}@staff.bond.edu.au

Abstract. Wireless ad hoc networks face many challenges in routing, power management, and basic connectivity. Existing research has looked into using predicted node movement as a means to improve connectivity. While past research has focused on assuming wireless signals propagate in clear free loss space, our previous research has focused on using signal loss maps to improve predictions. This paper presents novel testing of signal loss maps in relation to the accuracy used for prediction purposes. Through analysis of test cases and results from a custom built simulator the performance is effectively measured.

Keywords: Signal loss maps, predicting signal loss, signal strength maps.

1 Introduction

In any wireless networking environment where nodes are utilizing predicted location information in order to improve routing, the ability to predict the communication strength from one location to another becomes vital. A prediction of the wireless connectivity between two nodes cannot accurately rely on location information alone. The majority of today's wireless networking environments feature many obstructions which reflect and block wireless signals. This rules out assuming all signals between two physical locations will travel with a clear line-of-sight approach, such as is relied on in [1]. In addition to needing knowledge about nodes' physical locations, the predicted propagation of signals over physical areas is required. Wireless transmission capabilities in an unknown or known environment are difficult to predict with accuracy [2]. Not only do different locations have different communication capabilities, but environmental conditions may change those capabilities over time. Foreign nodes operating on the same channel, radio-frequency interference, environmental noise and even landscape may change radically over time. This will affect any recorded or estimated measurements of signal loss.

To overcome these challenges, our previous work designed a signal loss map solution [3]. Signal loss maps represent the logical signal propagation topology over a physical area. They describe how signals are likely to propagate in various directions over various distances. Due to the constantly changing nature of the wireless environment, a perfect signal loss map is not possible to create with current technologies. However, various estimates may be developed to provide, with appropriate safety margins, predictions on whether two nodes at two locations will have connectivity in the future.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 609 – 618, 2005.
© IFIP International Federation for Information Processing 2005

This signal loss map, dubbed the Communication Map (CM), is tailored to be built in real time using only wireless ad hoc nodes. The map is created using signal strength information provided with each packet as it is received from any node. To provide a physical reference system, some form of location-providing device is required for each node. In this research, a system such as GPS [4] is assumed to be available to provide the coordinates of each node. From these two external sources the CM is constructed.

The CM is made up of cells, defined areas which are square in shape and represent an average signal loss modifier. The signal loss modifier is a value which represents how a signal's loss increases over distance, relative to free space loss. This approach of using cells describes to users of the CM the same information that vendors of wireless cards use to describe range and signal strength capabilities. Vendors of wireless cards often include the maximum range and signal strength of their product in a variety of general scenarios, for example outdoors, home environment, cluttered office, etc. In a similar fashion, the CM of this research generates such scenarios in real time, and delineates where on a map such areas exist.

The general formula for calculating free space loss, S_{loss} (db), in ideal circumstances of wireless signals [5] is:

$$S_{loss} = 32.4 + 20Log_{10}F + 20Log_{10}D \tag{1}$$

where F is the frequency (MHz), and D is the distance (km) between the two nodes.

The free space loss formula may only be used *once* for the entire signal. To have an effect on the overall loss of a signal, each line *segment's* distance (how much a signal travels within a cell) will need to be adjusted by some modifier which represents a cell's effect on signal loss. To overcome this problem, the concept of *logical distance* is introduced. Any signal received or predicted using our CM is based on a *logical distance*. The *logical distance* that a signal travels is the distance it would need to physically travel in order to produce the same amount of loss, thus allowing the above formula to be used, given multiple signal modifiers.

Each cell represents the average signal loss of signals passing through that cell. The value stored for each cell is the modifier that a signal applies to the *physical distance* of a signal as it passes through that cell, which when multiplied together with the physical distance creates a *logical distance*. The minimum modifier value is 1.0, in other words a *logical distance* is identical to the *physical distance*, and thus represents perfect free-space loss. The modifier of each cell is used to extend the distance of a signal to the distance it would need to travel in perfect free space to achieve the same loss.

For an example of this process, consider Fig. 1 below. A signal will travel from Node A to Node B over the given CM. The direct line between the two nodes is formed, and a list of cells over which the signal will pass is created. Each of these cells multiply their modifier by the *physical distance* that the signal travels through their cell. The resulting *logical distance* can be used to find the signal loss, which can consequentially determine whether two nodes are predicted to be neighbours. This formula thus uses the modifier of each cell to *extend* the distance of a signal to the distance it would need to travel in perfect free space to achieve the same loss.

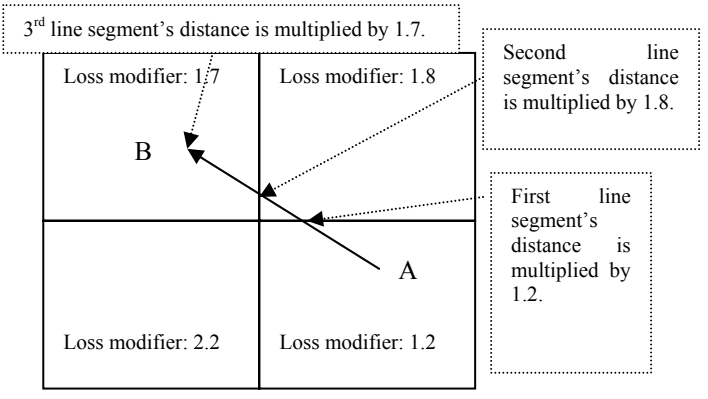


Fig. 1. Signal divided over multiple cells

2 Test Architecture

Existing simulators [6][7][8][9] already exist for basic network simulation. While many of these simulators are extensible, none specifically address the issues of wireless signal mapping and signal loss map testing. Because of this, a custom simulator was created. The simulator was created in response to fulfil the need for an appropriate testing bed for wireless ad hoc protocols that relied specifically on the needs of testing signal loss maps.

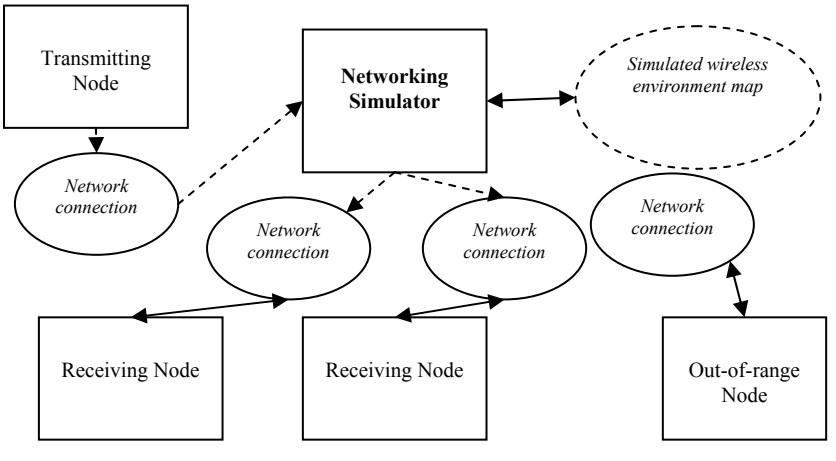


Fig. 2. Example Packet Broadcast

The simulator routes packets between nodes based on a simulated wireless environment map, which details user-created scenarios of how signals will propagate over real-world simulated objects. As each packet is successfully transmitted or lost (based on signal propagation), the simulator records what each node's CM predicted

the signal loss to be, along with the actual signal loss based on the simulated wireless map. Fig. 2 below shows how a packet would be routed from a transmitting node to nodes within range, based on the simulated wireless environment map for the given scenario.

3 Scenarios

Several scenarios have been created to validate the concept of the Communication Map, while also identifying its weaknesses. The field of wireless communication has an unlimited number of practical scenarios. This presents a significant challenge in representing a broad spectrum of possible scenarios to gauge the overall effectiveness of the CM. A total of seven scenarios have been designed to both represent realistic environments and test the various aspects of the algorithms presented in this paper and in our previous work [3]. Some scenarios have been tested with a varying number of nodes to further analyse scenarios while also gaining an insight into the effects of network population.

Scenario 6 is presented as an example scenario. It contains only four nodes with two simulated wireless propagation areas. To the left is an area with a small signal loss modifier, such as that found in a forest. On the right-hand side is a building with reasonably thick walls (having a logical distance of 400 meters) and a reasonably high signal loss modifier of 5.0. All four nodes are in motion in this scenario, with nodes C and D moving from outside the building over to the forest area, and nodes A and B moving within the building. The scenario diagram is included in Fig. 3.

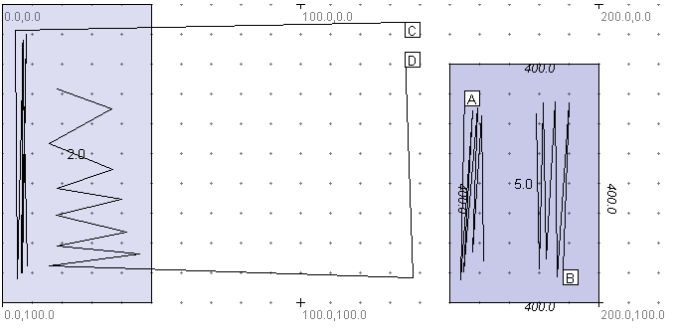


Fig. 3. Scenario 6 Overview

4 Results

Each of the seven scenarios were executed for each variation in settings and for each variation in node population where the scenario allowed multiple node populations. The results were then processed and summarised across different combinations of settings so that well-informed analysis could be conducted.

4.1 Boundaries and Default Cell Size (DCS)

The choice of implementation between boundaries and the basic CM is significant. The use of boundaries is a modification to the original algorithm as an attempt to provide possible improvement to the CM solution. With boundaries, each cell not only contains an average signal loss modifier for the area within the cell, but also additional modifiers for signals entering or leaving the cell through any of the 4 borders of the cell. The theory behind this was that certain real-world structures, such as buildings, had a far higher signal loss, but were not based over an area but rather a single object passed through between areas.

In the majority of these tests, two sets of results are generated for each scenario. One set is generated using the basic algorithm without boundaries, and one set is generated using the algorithm implementing boundaries. In each set, three Default Cell Size (DCS) settings are implemented, 25 meters, 50 meters, and 100 meters (the default). This is done to determine how reducing the DCS affects accuracy. In theory, the smaller the DCS, the more similar the results should be to a generic boundary implementation. This is due to the fact that a smaller DCS enables the Communication Map algorithms to more accurately map signal loss immediately. With larger DCS settings, the loss accounted for in artificial boundaries is averaged into the larger cells.

The first scenario tested for boundaries is the Scenario 4. This scenario was created specifically to test the boundary concept. The scenario consists of a single building, with perfect free-space loss within the building (for example in an empty warehouse) with basic walls of 150 logical meter loss (about that of a thin wooden wall). 3 nodes are placed within the building, with a further 5 nodes outside it. Two of the nodes, E and C, circumnavigate the building.

The results from these experiments are shown in Fig. 4 and Fig. 5. Without implementing boundaries, the average error stabilises between 35% and 50% using a DCS of 100 meters. Reducing the DCS by half improves accuracy considerably, with further reduction possible if the DCS is lowered to 25 meters. This demonstrates that a reduced DCS allows the CM to map some of the effects of simulated boundaries as implementing the boundaries algorithm does (when compared with Fig. 5). Implementing boundaries shows significantly improved accuracy, though more than this experiment alone is required to prove this.

Interesting to note that reducing the DCS does not improve accuracy in Scenario 4 with boundaries, but has the reverse affect. Fig. 6 - Fig. 9 illustrate the CMs at the end

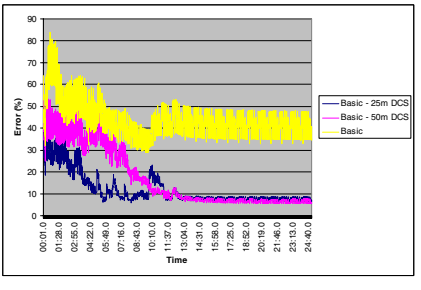


Fig. 4. Scenario 4 without Boundaries

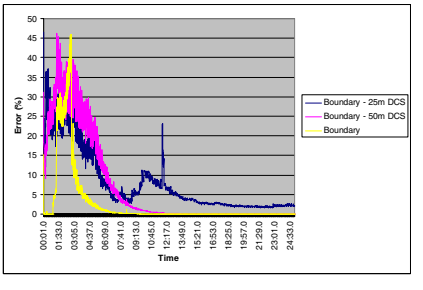


Fig. 5. Scenario 4 with Boundaries

of each scenario from Node A's viewpoint. Fig. 6 shows how the CM algorithm represented the signal loss surrounding nodes A, B, and F by utilising the smaller cell size of subdividing the DCS as a boundary representation itself. Fig. 7 shows the typical 100m DCS with boundaries, with Fig. 8 and Fig. 9 reducing the DCS to 50m and 25m respectively. The smaller DCS with boundaries hinders the boundary development, as boundaries are more difficult to develop than cells. Fig. 6 without boundaries shows that even with 8 nodes the CM does not perfectly represent even the shape of the simulated building, let alone the accurate signal loss. Without a thorough grid of cells there will be varying quantities of signals mapped over various cells, simply from node positioning and node movement. A perfectly accurate CM is impossible to develop using current technology. With the smaller DCS on the boundary examples, there become too many objects where signals can be mapped. As signals being mapped are averaged over all objects along the assumed signal's path, the greater number of objects requires a greater quantity of node movement to further correlate signal loss to more accurately-placed objects.

To further see the effects of implementing boundaries more scenarios are needed. The same experiments were run on Scenario 5 and Scenario 6, both of which also have high boundary use yet simple simulated environments that have a better chance

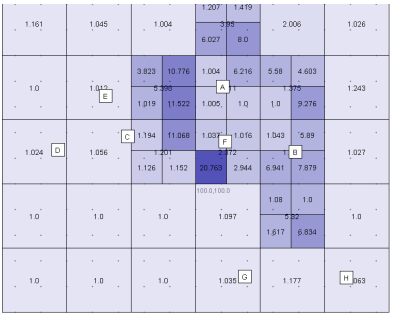


Fig. 6. Scenario 4: 25m DCS without Boundaries

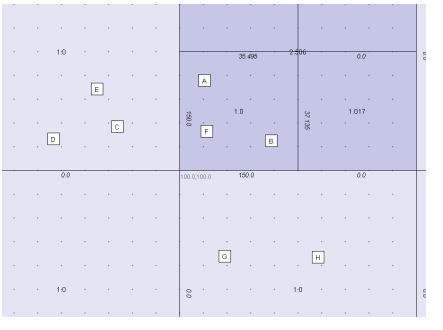


Fig. 7. Scenario 4: 100m DCS with Boundaries

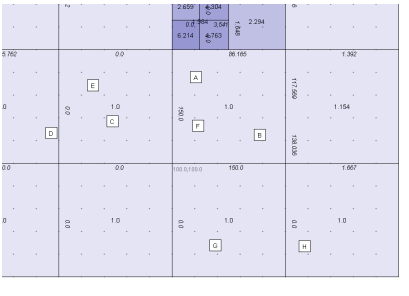


Fig. 8. Scenario 4: 50m DCS with Boundaries

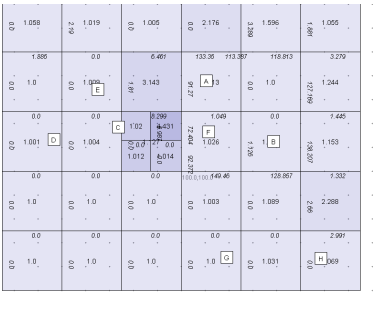


Fig. 9. Scenario 4: 25m DCS with Boundaries

of being mapped. Results are shown in Fig. 10 and Fig. 11. These experiments indicate that overall the varying DCS settings and the implementation of boundaries make little difference to the overall accuracy.

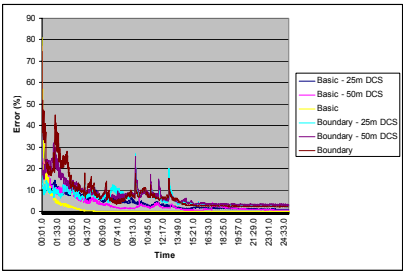


Fig. 10. Scenario 5 Results

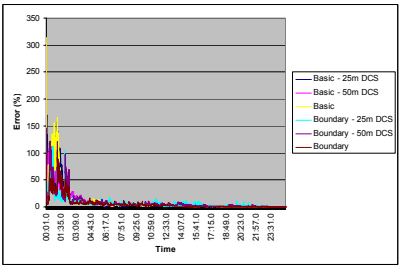


Fig. 11. Scenario 6 Results

The experiments were then run on Scenario 7, which is based on a real street. Despite the number of simulated objects and boundaries, all settings produced similar results (refer to Fig. 12 and Fig. 13). The CMs at the end of the simulations of a 25m DCS without boundaries and a 100m DCS with boundaries experiment are shown in Fig. 14 and Fig. 15 for interest. The favourable results in these experiments can be

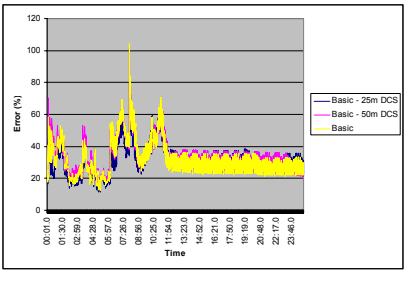


Fig. 12. Scenario 7 without Boundaries

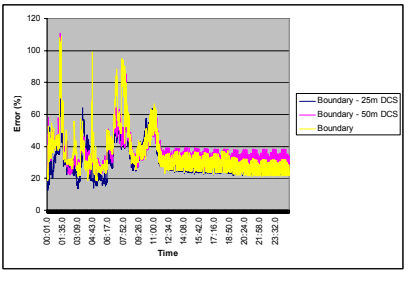


Fig. 13. Scenario 7 with Boundaries

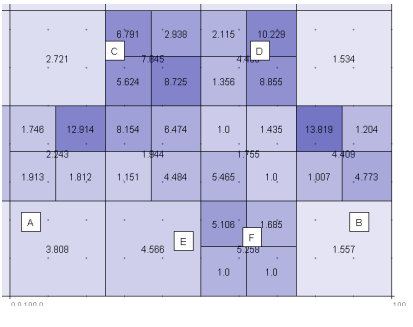


Fig. 14. 25m DCS CM without Boundaries

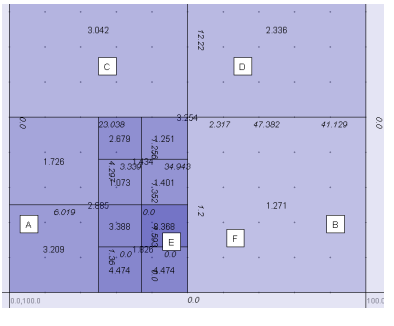


Fig. 15. 100m DCS CM with Boundaries

attributed to the low amount of node movement, where only three nodes are moving in relatively fixed movement patterns. A perfect CM is not required to produce favourable results, only a CM which adequately represents the signal loss as it has and will be used in the future.

4.2 Number of Nodes

Another area of interest was to determine if the number of nodes had an effect of the accuracy of the CM. Fig. 16 and Fig. 17 graph all setting samples over Scenario 1 and Scenario 2 respectively. These are the two main scenarios where the basic layout would allow the number of nodes to play an influential role without new areas being discovered by the increased number of nodes. Scenarios 3 and 4 were included (Fig. 18 and Fig. 19) were tested also, though these examples focus mainly on boundary testing.

The results in these experiments were surprisingly different than anticipated. In theory, the number of nodes would have favourably increased the accuracy of the CM. However, in almost all scenarios tested, an increase in the number of nodes had an adverse effect on accuracy. In Fig. 16, using both 4 and 5 nodes resulted in very similar results, with 5 nodes performing slightly better, as would be expected. Using 6 nodes, however, almost doubled the overall average error.

In Scenario 2 (Fig. 17), 8 moving nodes generate a more accurate map than 6 moving nodes, but only within the first 10 minutes of the experiment. Towards the end of the simulation the average error increases, and unfortunately stabilises this way due to the lack of future node movement. This is simply a case of a new area of signal

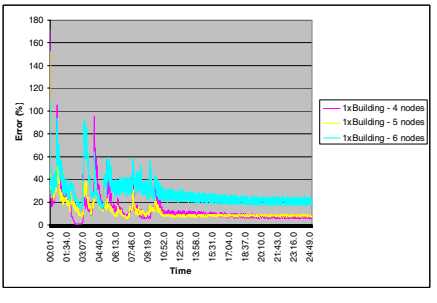


Fig. 16. Scenario 1

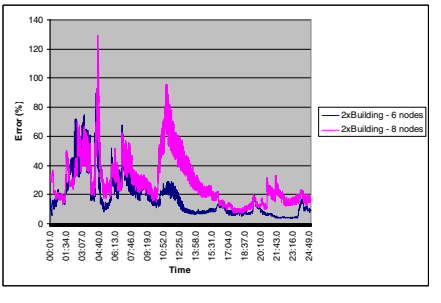


Fig. 17. Scenario 2

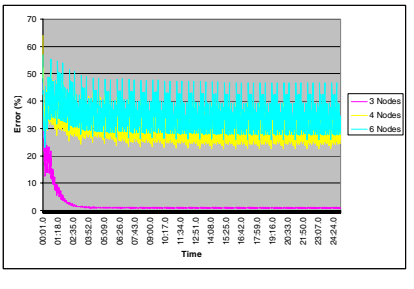


Fig. 18. Scenario 3

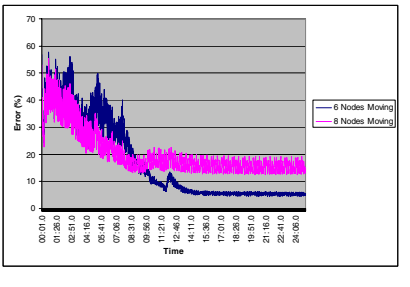


Fig. 19. Scenario 4

loss being discovered, where a lack of node movement fails to accurately identify that loss. In Scenario 3 (Fig. 18), the best performance is surprisingly from using only 3 nodes. While the actual CM built using only 3 nodes is far from being an accurate representation of the simulated signal loss, it provides enough detail given the information the nodes have accumulated.

Both in theory and in the experiments performed, a greater number of nodes has a greater chance of producing a more accurate CM. However, actual prediction accuracy depends largely on how the CM is used by nodes during the scenarios. Nodes which remain in the same approximate area both further improve the accuracy of that area and benefit from the effects. Nodes which use the CM in untested areas are less likely to obtain ideal results. Due to the vast nature of exact node movement and positioning, various experiments will achieve various results. Overall, however, a more accurate CM is created given a greater number of nodes.

4.3 MCSD and MCMD

The Minimum Cell Subdivide Difference (MCSD) and Minimum Cell Merge Difference (MCMD) values have also been considered. These values govern when cells are able to subdivide (split into 4 equal parts) and merge back together when the level of detail required by the CM changes over time. By default, all scenarios use a MCSD of 3.0 and a MCMD of 2.0. This means that if the average modifier within a cell varies by more than 3.0 then the cell will subdivide to allow for a greater level of detail to be mapped. If the average modifier between 4 adjacent nodes falls to less than 2.0, then the cells are merged back together again. These settings have been modified in two groups of settings; the first uses a MCSD of 2.0 and a MCMD of 1.0, and the second uses a MCSD of 1.0 and a MCMD of 0.0. These are all compared with basic settings in Fig. 20 below.

The results from these experiments show that lowering the MCSD and MCMD values increases accuracy. The reasoning behind this is that lower MCSD and MCMD

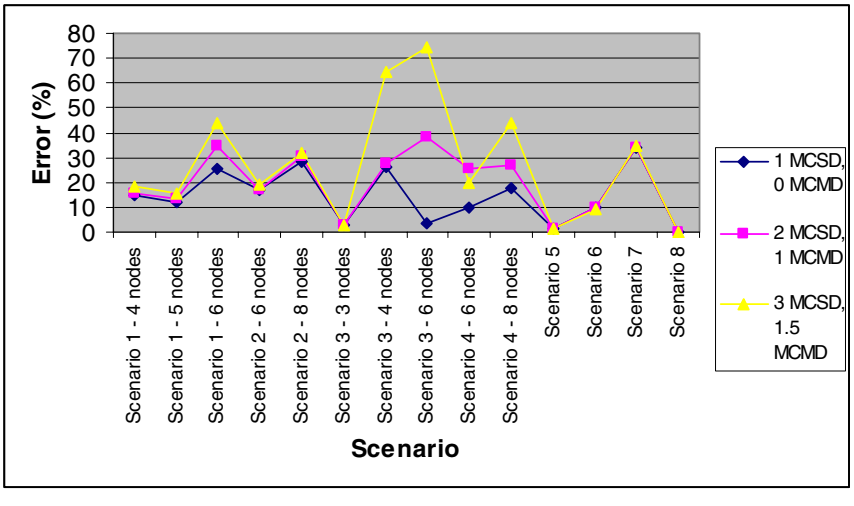


Fig. 20. MCSD and MCMD Performance

values allow cells to subdivide faster and merge back together without difficulty. As previously discussed with DCS values, the smaller the cells the more accuracy is obtained, and lowering the MCSD and MCMD values have the same effect. However, a reduced DCS value increases bandwidth costs.

The scenarios focusing specifically on boundary-optimised situations benefit the most from lower MCSD and MCMD values, as smaller cells more closely represent boundaries. The most interesting results are that the more realistic scenarios (Scenario 2, Scenario 6, and Scenario 7) show almost no difference in changing the MCSD and MCMD. From this it is concluded that while the MCSD and MCMD values in theory have an effect on accuracy, in practice the Communication Map algorithms perform well regardless of these settings.

5 Conclusion

Predicting connectivity between nodes based on location information can be improved with an understanding of wireless signal propagation in each environment. Signal loss maps generate this information, and our Communication Map solution achieves this in real time without user intervention. The use of signal loss maps in wireless ad hoc routing is a relatively new field, and one which requires significant performance analysis before the advantages become evident. This paper has presented such testing under a number of custom-created scenarios, as well as the various settings which may be used to improve accuracy. The results of these tests conclude that signal loss maps can perform accurately under a variety of situations. Further testing can be conducted if the CM is applied to existing routing protocols, and performance investigated.

References

1. Su, W. W., Motion Prediction in Mobile/Wireless Networks. PhD dissertation. University of California, Los Angeles, USA. 2000.
2. Howard, A., Siddiqi, S., Sukatme, G. S., An Experimental Study of Localization Using Wireless Ethernet, in 4th International Conference on Field and Service Robotics, July 2003.
3. Larkin, H., Wireless Signal Strength Topology Maps in Mobile Adhoc Networks, Embedded and Ubiquitous Computing, International Conference EUC2004, pp. 538-547, Japan. 2004.
4. Enge, P., Misra, P. Special issue on GPS: The Global Positioning System. Proceedings of the IEEE, pages 3–172, January 1999.
5. Shankar, P. M., Introduction to Wireless Systems. Wiley, USA. 2001.
6. Fall, K. Network Emulation in the Vint/NS Simulator. Proceedings of ISCC'99, Egypt, 1999.
7. McDonald, C.S., A Network Specification Language and Execution Environment for Undergraduate Teaching. Proceedings of the ACM Computer Science Education Technical Symposium '91, San Antonio, pp25-34, Texas, Mar 1991.
8. Unger, B., Arlitt, M., et. al., ATM-TN System Design. WurcNet Inc. Technical Report, September 1994.
9. Keshav, S. REAL: A Network Simulator, tech. report 88/472, University of California, Berkeley, 1988.

Performance Analysis of Adaptive Mobility Management in Wireless Networks

Myung-Kyu Yi

Dept. of Computer Science & Engineering Korea University,
1,5-Ga, Anam-Dong, SungBuk-Gu, Seoul 136-713, South Korea
kainos@disys.korea.ac.kr

Abstract. In this paper, we propose an adaptive mobility management scheme for minimizing signaling costs in Hierarchical Mobile IPv6 (HMIPv6) networks. In our proposal, if the mobile node's mobility is not local, the mobile node sends location update messages to correspondent nodes in the same way as Mobile IPv6 (MIPv6). After the creation of a spatial locality of the mobile node's movement, the mobile node sends location update messages to the correspondent nodes in same way as HMIPv6. Therefore, our proposal can reduce signaling costs in terms of packet transmission delays in HMIPv6 networks. The cost analysis presented in this paper shows that our proposal offers considerable performance advantages to MIPv6 and HMIPv6.

1 Introduction

MIPv6 allows an IPv6 node to be mobile to arbitrarily change its location on the IPv6 Internet and still maintain existing connections. However, MIPv6 causes in a high signaling cost when it updates the location of an Mobile Node (MN) if it moves frequently. Thus, HMIPv6 is proposed by IETF to reduce signaling costs. It is well known that the performance of HMIPv6 is better than that of MIPv6. This is especially true when the basic assumption is that 69% of a user's mobility is local. If the user's mobility is not local, performance of HMIPv6, in terms of delays for packet delivery, is worse than that of MIPv6, due to the encapsulation processing by the Mobility Anchor Point (MAP). Since all packets from a CN to an MN are first delivered through the MAP, it is possible that the MAP can become bottlenecked. Therefore, the load of the search and tunnelling processes increase on the MAP as the number of MNs increase in the foreign or home networks. It is a critical problem for the performance of HMIPv6 networks. In this paper, therefore, we propose an adaptive mobility management scheme, called AHMIPv6, for minimizing signaling costs in HMIPv6 networks. In our proposal, an MN sends a Binding Update (BU) message to the Correspondent Nodes (CN) with either its on-link address (LCoA) or Regional Care of Address (RCoA), depending on the MN's mobility pattern. Thus, our proposal can reduce signaling loads, even if a user's mobility is not local.

The rest of the paper is organized as follows. Section 2 describes the proposed procedures of location update and packet delivery using the adaptive mobility

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 619–628, 2005.
© IFIP International Federation for Information Processing 2005

management scheme called AHMIPv6. In Section 3, we propose the analytic mobility model based on the random walk. Section 4 formulates the location update cost and the packet delivery cost using the analytic model. Section 5 shows the evaluation of the proposed system's performance and analysis of the results. Finally, our conclusions are presented in Section 6.

2 Adaptive Mobility Management Scheme

This section describes the location update and packet delivery procedure. In our proposal, each MN has a value of m and T_m. While m is the number of subnet crossings within the MAP domain, T_m is the threshold value which decides whether the MN sends a BU message to the CN with the LCoA or the RCoA. Whenever an MN enters a new MAP domain, it sets the value of m to zero. T_m can be adjusted based on the user's mobility pattern and current traffic load.

The procedures for a location update are as follows:

- If an MN moves to a different MAP domain, then:

1) the MN obtains two CoAs: an LCoA and an RCoA.
2) Then, it registers with its MAP and HA by sending a BU message, and it sets the value of m to zero.

- Otherwise, if an MN moves within the same MAP domain, then:

1) the MN gets a new LCoA.
2) The MN registers with its MAP by sending a BU message.

After registration with the MAP, the MN compares the value of m with T_m.

Case 1. If the value of m is less than T_m, then:

3-1) the MN sends a $BU_{[HoA,LCoA]}$[1] message to the CN.

Case 2. Otherwise, if the value of m is greater than or equal to T_m, then:

3-2) the MN sends a $BU_{[HoA,RCoA]}$[2] message to the CN. After the sending of the $BU_{[HoA,RCoA]}$ message, the MN does not send any other BU messages to the CN before it moves out of the MAP domain.

As a result, the MN performs registration with the CNs using an RCoA or an LCoA, depending on its mobility pattern. In our proposal, the packet delivery procedure is exactly the same to that in MIPv6 or HMIPv6.

[1] BU with the binding between the MN's Home Address (HoA) and LCoA.
[2] BU with the binding between the MN's Home Address (HoA) and RCoA.

3 Analytic Mobility Model

Inspired by the initial idea in [1,2], we will describe a two-dimensional random walk model for mesh planes. Our model is similar to [1] and considers a regular MAP domain/subnet overlay structure. In this model, the subnets are grouped into several n-layer MAP domains. Every MAP domain covers $N = 4n^2 - 4n + 1$ subnets.

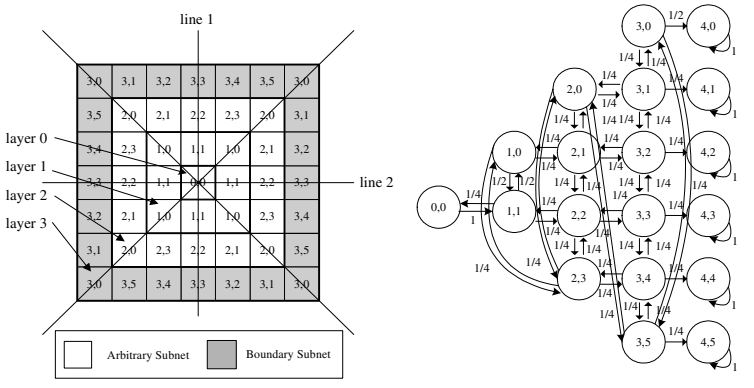


Fig. 1. Type Assignments of the mesh 4-layer MAP domain and the State Diagram

As shown in Fig. 1 (where n = 4), the subnet at the center of a MAP domain is called *layer 0*. An n-layer MAP domain consists of a subnet from layer 0 to layer $n-1$. Based on this domain/subnet structure, we derive the number of subnet crossings before an MN crosses a MAP domain boundary. According to the equal moving probability assumption (i.e., with a probability of 1/n), the subnets in a MAP domain are classified into several subnet types, based on the type classification algorithm in [2]. A subnet type is of the form $< x, y >$, where x indicates that the subnet is in layer x and y represents the $y+1$st type in layer x. Based on the type classification and the concept of absorbing states, the state diagram of the random walk for an n-layer MAP domain is shown in Fig. 1. In this state diagram, state (x, y) represents that the MN is in one of the MAP domains of type $< x, y >$, where the scope of x and y is

$$0 \le x \le n, \quad \begin{cases} 0 \le y \le 2x-1 & ,if \quad x \ge 1 \\ y = 0 & ,if \quad x = 0. \end{cases} \tag{1}$$

State (n, y) represents that the MN moves out of the MAP domain from state $(n-1, y)$, where $0 \le y \le 2n-3$. For $x = n$ and $0 \le y \le 2n-3$, the states (n, y) are absorbing, and the others are transient. For $n > 1$, the total number $S(n)$ of states for the n-layer MAP domain random walk is $n^2 + n - 1$. The transition matrix of this random walk is an $S(n) \times S(n)$ matrix $P = (p_{(x,y)(x',y')})$. Therefore, $P = (p_{(x,y)(x',y')})$ can be defined as the one-stop transition probability from state (x, y) to state (x', y') (i.e., which represents the probability that the MN moves

from a $< x, y >$ subnet to a $< x', y' >$ subnet in one step). We use the Chapman-Kolmogorov equation to compute $p^{(r)}_{(x,y)(x',y')}$, which is the probability that the random walk moves from state (x, y) to state (x', y') with exact r steps. We define $p_{r,(x,y)(n,j)}$ as the probability that an MN initially resides at a $< x, y >$ subnet, moves into a $< n-1, j >$ subnet at the $r-1$ step, and then moves out of the MAP domain at the r step as follows:

$$p_{r,(x,y)(n,y)} = \begin{cases} p_{(x,y)(n,y)} & , for \quad r = 1 \\ p^{(r)}_{(x,y)(n,y)} - p^{(r-1)}_{(x,y)(n,y)} & , for \quad r > 1. \end{cases} \tag{2}$$

4 Signaling Cost Functions

To investigate the performance of MIPv6, HMIPv6 and AHMIPv6, the total signaling costs given to the HA, CN, and MAP to handle mobility of the MNs are analyzed. We assume that the performance metric is the total signaling cost. It consists of the location update cost and packet delivery cost.

4.1 Mobile IPv6

Location Update Cost in MIPv6. We define the costs and parameters used for the performance evaluation of the location update as follows:

- U_{HA} : The location update cost of a BU for the HA
- U_{CN} : The location update cost of a BU for the CN
- U_{MAP} : The location update cost of a BU for the MAP
- u_{hn} : The transmission cost of a BU between the HA and the MN
- u_{cn} : The transmission cost of a BU between the CN and the MN
- u_{mn} : The transmission cost of a BU between the MAP and the MN
- a_h : The processing cost of a location update at the HA
- a_m : The processing cost of a location update at the MAP
- l_{hn} : The average distance between the HA and the MN
- l_{cn} : The average distance between the CN and the MN
- l_{hm} : The average distance between the HA and the MAP
- l_{mn} : The average distance between the MAP and the MN
- δ_U : The proportionality constant for a location update.

According to the signaling message flows for the BU, each cost of location update can be calculated as follows:

$$U_{HA} = a_h + 2u_{hn}, \quad U_{CN} = u_{cn}, \quad U_{MAP} = a_m + 2u_{mn} \tag{3}$$

For simplicity, we assume that the transmission cost is proportional to the distance, in terms of the number of hops between the source and destination mobility agents, such as the HA, MAP, CN and MN. Using the proportional constant δ_U, each cost of location update can be rewritten as follows:

$$U_{HA} = a_h + 2l_{hn} \cdot \delta_U, \quad U_{CN} = l_{nc} \cdot \delta_U, \quad U_{MAP} = a_m + 2l_{mn} \cdot \delta_U \tag{4}$$

We derive the number of subnet crossings and location updates between the beginning of one session and the beginning of the next session before an MN leaves the first MAP domain. Similar to [1], we define the additional costs and parameters used for the performance evaluation of the location update as follows.

- r : The number of the MN's subnet crossings
- d : The number of subnet crossings before an MN leaves the first MAP domain
- t_d : The time interval between the beginning of one session and the beginning of the next session
- $r(t_d)$: The number of the MN's subnet crossings during t_d
- l : The number of subnet crossings before an MN leaves the first MAP domain during t_d
- N : The total number of subnets within a MAP domain
- $1/\lambda_m$: The expected value for the subnet residence time
- $1/\lambda_d$: The expected value for the t_d distribution
- σ : The number of the CNs that have a binding cache for the MN

We assume that an MN is in any subnet of a MAP domain with equal probability. This implies that the MN is in subnet $< 0, 0 >$ with a probability of $1/N$ and is in a subnet of type $< x, y >$ with a probability of $4/N$, where $N = 4n^2 - 4n + 1$ is the number of subnets covered by an n-layer MAP domain. From (2), we derive d as the number of subnet crossings before an MN leaves the first MAP domain as follows:

$$d = \frac{1}{N}\left(\sum_{k=1}^{\infty} k \cdot \sum_{j=0}^{2n-3} p_{k,(0,0)(n,j)}\right) + \frac{4}{N}\left(\sum_{k=1}^{\infty} k \cdot \sum_{x=0}^{n-1}\sum_{y=0}^{2x-1}\sum_{j=0}^{2n-3} p_{k,(x,y)(n,j)}\right) \quad (5)$$

We denote $\alpha(r)$ as the probability that an MN will leave the MAP domain at the rth step provided that the MN is initially in an arbitrary subnet of the MAP domain as follows:

$$\alpha(r) = \frac{1}{N}\left(\sum_{j=0}^{2n-3} p_{r,(0,0)(n,j)}\right) + \frac{4}{N}\left(\sum_{x=1}^{n-1}\sum_{y=0}^{2x-1}\sum_{j=0}^{2n-3} p_{r,(x,y)(n,j)}\right) \quad (6)$$

Note that the above derivations are based on the equal moving probability assumption, thus, we derive the number of subnet updates between the beginning of one session and the beginning of the next session. Fig. 2 shows the timing diagram of the activities for an MN. Assume that the previous session of the MN begins at time t_0, and the next session begins at time t_1. Let $t_d = t_1 - t_0$, which has a general distribution with a density function of $f_d(t_d)$, and expect a value of $1/\lambda_d$, and Laplace Transform

$$f_d^*(s) = \int_{t_d=0}^{\infty} e^{-st_d} f_d(t_d) dt_d \quad (7)$$

We denote $r(t_d)$ as the number of location updates for the MAP during the period t_d. Since an MN needs to register with the MAP whenever it moves in

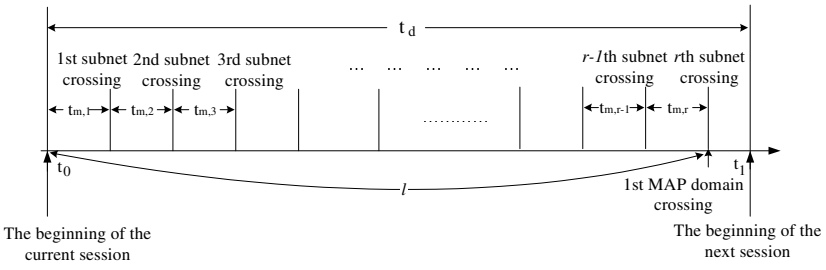


Fig. 2. Time diagram for subnet crossings

HMIPv6, $r(t_d)$ is equal to the number of subnet crossings during t_d. Assume that the subnet residence time $t_{m,j}$ at j-th subnet has an Erlang distribution, with a mean of $1/\lambda_m = m/\lambda$, variance $V_m = m/\lambda^2$, and density function as follows:

$$f_m(t) = \frac{\lambda e^{-\lambda t}(\lambda t)^{m-1}}{(m-1)!} \ (where \ \ m = 1, 2, 3, \cdots) \tag{8}$$

Notice that an Erlang distribution is a special case of the Gamma distribution, where the shape parameter m is a positive integer. Since the subnet crossings of an MN can be modelled as an equilibrium Erlang-renewal process, we can get the probability mass function of the number of subnet crossings $r(t_d)$ within t_d from (7) and (8) as follows:

$$Pr[r(t_d) = k] = \int_{t_d=0}^{\infty} \frac{e^{-\lambda t_d}}{m} \tag{9}$$
$$\times \Big(\sum_{j=km}^{km+m-1} \left[\frac{(km+m-j)(\lambda t_d)^j}{j!} \right] - \sum_{j=km-m}^{km-1} \left[\frac{(j-km+m)(\lambda t_d)^j}{j!} \right] \Big) \cdot f_d(t_d) \ dt_d$$
$$(where \ k = 1, 2, \cdots)$$
$$Pr[r(t_d) = 0] = \int_{t_d=0}^{\infty} \frac{e^{-\lambda t_d}}{m} \sum_{j=0}^{m-1} \left[\frac{(m-j)(\lambda t_d)^j}{j!} \right] \cdot f_d(t_d) \ dt_d$$

We denote l as the number of subnet crossings before an MN leaves the first domain during t_d. From (5) and (9), we can get $D(t_d)$ as follows:

$$l = \frac{1}{N} \left(\sum_{k=1}^{l} k \cdot \sum_{j=0}^{2n-3} p_{k,(0,0)(n,y)} \cdot Pr[r(t_d) = k] \right) \tag{10}$$
$$+ \frac{4}{N} \left(\sum_{k=1}^{l} k \cdot \sum_{x=0}^{n-1} \sum_{y=0}^{2x-1} \sum_{j=0}^{2n-3} p_{k,(x,y)(n,j)} \cdot Pr[r(t_d) = k] \right)$$

In MIPv6, an MN sends a BU message whenever it changes its point of attachment transparently to the IPv6 networks. From (4)-(10), we can get the total location update cost before an MN leaves the first MAP domain during t_d in MIPv6 as follows:

$$U = \left(U_{HA} + \sigma \cdot U_{CN}\right) \cdot \sum_{j=0}^{l} \left(j \cdot Pr[r(t_d) = j] \cdot \alpha(j)\right) \tag{11}$$

Packet Delivery Cost in MIPv6. The packet delivery cost consists of transmission and processing costs. First of all, we define the additional costs and parameters used for the performance evaluation of the packet delivery cost as follows:

- f_{ch} : The transmission cost between the CN and the HA
- f_{hn} : The transmission cost between the HA and the MN
- f_{cn} : The transmission cost between the CN and the MN
- l_{ch} : The average distance between the CN and the HA
- v_h : The processing cost of the packet delivery at the HA
- $E(S)$: The average session size in the unit of the packet
- λ_α : The packet arrival rate for each MN
- δ_D : The proportionality constant for the packet delivery
- δ_h : The packet delivery processing cost constant at the HA

In MIPv6, route optimization is used to resolve the triangular routing problem. Thus, only the first packet of a session transits to the HA to detect whether or not an MN moves into foreign networks. Subsequently, all successive packets of the session are directly routed to the MN. As a result, the packet delivery cost during t_d can be expressed as follows:

$$F = \lambda_\alpha(f_{ch} + f_{hn} + (E(S) - 1)f_{cn}) + \lambda_\alpha \delta_h \tag{12}$$

We assume that the transmission cost of the packet delivery is proportional to the distance between the sending and receiving mobility agents, with the proportionality constant δ_D. Therefore, f_{ch}, f_{hn}, and f_{cn} can be represented as $f_{ch}{=}l_{ch}\delta_D$, $f_{hn}{=}l_{hn}\delta_D$, and $f_{cn}{=}l_{cn}\delta_D$. Also, we define the proportionality constant as δ_h. The δ_h is a packet delivery processing constant for the lookup time of a binding cache at the HA. Therefore, v_h can be represented as $v_h{=}\lambda_\alpha\delta_h$. Finally, we can get the packet delivery cost during t_d as follows:

$$F = \lambda_\alpha\left(l_{ch} + l_{hn} + (E(S) - 1)l_{cn}\right)\delta_D + \lambda_\alpha \delta_h \tag{13}$$

Based on the above analysis, we get the total signaling cost function in MIPv6 from (11) and (13):

$$C_{MIP}(\lambda_m, \lambda_d, \lambda_\alpha) = U + F \tag{14}$$

4.2 Hierarchical Mobile IPv6

To investigate the performance of HMIPv6, we define the additional costs and parameters used for the performance evaluation of location updates as follows.

- l_{cm} : The average distance between the CN and the MAP
- δ_m : The packet delivery processing cost constant at the MAP

In HMIPv6, an MN sends a BU message to the MAP whenever it moves within the MAP domain after the registration with the HA. From (4) and (11), we can get the total location update cost before an MN leaves the first MAP domain during t_d in HMIPv6 as follows:

$$U^{'} = U_{HA} + \sigma \cdot U_{CN} + U_{MAP} \sum_{j=0}^{l} \big(j \cdot Pr[r(t_d) = j] \cdot \alpha(j)\big) \tag{15}$$

In HMIPv6, all packets destined for the MN are forwarded by the HA and the MAP using the encapsulation and decapsulation process. In a similar manner to HMIPv6, we define a proportionality constant of δ_m. The δ_m is a packet delivery processing constant for the lookup time of a binding cache at the MAP. From (12), we can get the total packet delivery cost during t_d in HMIPv6 as follows:

$$F^{'} = \lambda_\alpha \Big((l_{ch} + l_{hm} + l_{mn}) + (E(S) - 1)(l_{cm} + l_{mn})\Big)\delta_D + \lambda_\alpha(\delta_h + E(S)\delta_m) \tag{16}$$

Based on the above analysis, we get the total signaling cost function in HMIPv6 from (15) and (16) as follows:

$$C_{HMIP}(\lambda_m, \lambda_d, \lambda_\alpha) = U^{'} + F^{'} \tag{17}$$

4.3 Adaptive Hierarchical Mobile IPv6

The optimal value of T_m is defined as the value of l that minimizes the cost function derived in Section 4.1 and 4.2. To get the value of T_m, we define the cost difference function between MIPv6 and HMIPv6 as follows:

$$\Delta(l, \lambda_m, \lambda_d, \lambda_\alpha) = C_{MIP} - C_{HMIP} \tag{18}$$

Given Δ, the algorithm to find the optimal value of l is defined as follows:

$$T_m(\lambda_m, \lambda_d, \lambda_\alpha) = \begin{cases} 0 & , if\ \Delta(l, \lambda_m, \lambda_d, \lambda_\alpha) > 0 \\ maximum(l : \Delta(l, \lambda_m, \lambda_d, \lambda_\alpha) \leq 0) & , otherwise \end{cases} \tag{19}$$

The optimal value of T_m is a designed value. It is computed before the communications based on the average packet arrival rate λ_α, average mobility rate λ_m, and average λ_d. In the AHMIPv6, an MN sends the BU message to the CN with either an LCoA or an RCoA. If the number of subnet crossings, m, is less than T_m, the MN sends the BU message to the CN with an LCoA after the registration with the HA and MAP. Otherwise, the MN sends a BU message to the CN with an RCoA. From (11),(15) and (19), we can get the total location update cost before an MN leaves the first MAP domain during t_d in the AHMIPv6 as follows:

$$U^{''} = \begin{cases} U_{HA} + (U_{MAP} + \sigma \cdot U_{CN}) \sum_{j=0}^{T_m} \big(j \cdot Pr[r(t_d) = j]\alpha(j)\big), & if \quad T_m < j \\ U_{HA} + U_{MAP} \sum_{j=0}^{l} \big(j \cdot Pr[r(t_d) = j]\alpha(j)\big) + \sigma \cdot U_{CN} & \\ \times \sum_{j=0}^{T_m} \big(j \cdot Pr[r(t_d) = j]\alpha(j)\big) & , if \quad T_m \geq j \end{cases} \tag{20}$$

In the AHMIPv6, when a CN sends a packet to the MN, the packets are directly forwarded to the MN using an LCoA, if m is less than T_m. Otherwise, the packets are indirectly forwarded to the MN via the MAP using the RCoA. From (12) and (16), therefore, we can get the total packet delivery cost during t_d in AHMIPv6 as follows:

$$F^{''} = \begin{cases} F, & if \;\; T_m < j \\ F^{'}, & if \;\; T_m \geq j \end{cases} \tag{21}$$

Based on the above analysis, we get the total signaling cost function in AHMIPv6 from (20) and (21) as follows:

$$C_{AHMIP}(\lambda_m, \lambda_d, \lambda_\alpha) = U^{''} + F^{''} \tag{22}$$

5 Numerical Results

In this section, we will demonstrate some numerical results. Table 1 shows some of the parameters used in our performance analysis that are discussed in [1]. For simplicity, we assume that the distance between the mobility agents is fixed and has the same number of hops (i.e., $l_{ch} = l_{cm} = l_{hm} = l_{mn} = l_{cn} = 10$).

Table 1. Performance Analysis Parameters

Parameter	Value	Parameter	Value	Parameter	Value	Parameter	Value
N	49	L	28	α	1.5	κ	1-10
λ_α	0.01-10	λ_i	0.01-10	λ_m	0.01-10	λ_d	0.01-10
a_m	20	a_h	30	δ_m	10	δ_D	0.2
δ_U	15	δ_h	15	n	1-10	σ	1-10

Fig. 3 (a) and (b) show the effect of the mobility rate λ_m on the total signaling cost for $\lambda_\alpha = 1$ and $\lambda_d = 0.01$. As shown in Fig. 3 (a) and (b), the total signaling cost increases as the mobility rate λ_m increases. We can see that the performance of the AHMIPv6, on the whole, results in the lowest total signaling cost, compared with MIPv6 and HMIPv6. These results are expected because the AHMIPv6 scheme tries to reduce the signaling loads at the MAP for the small value of λ_m in the same way as MIPv6. For the large value of λ_m, the AHMIPv6 scheme tries to reduce the location update costs by sending a BU message to the CN with an RCoA in the same way as HMIPv6.

Fig. 3 (c) and (d) show the effect of the packet arrival rate λ_α on the total signaling cost for $\lambda_m = 1$ and $\lambda_d = 0.01$. As shown in Fig. 3 (c) and (d), the total signaling cost increases as the packet arrival rate λ_α increases. We can see that the performance of the AHMIPv6, on the whole, results in the lowest total signaling cost compared with MIPv6 and HMIPv6. From the above analysis of the

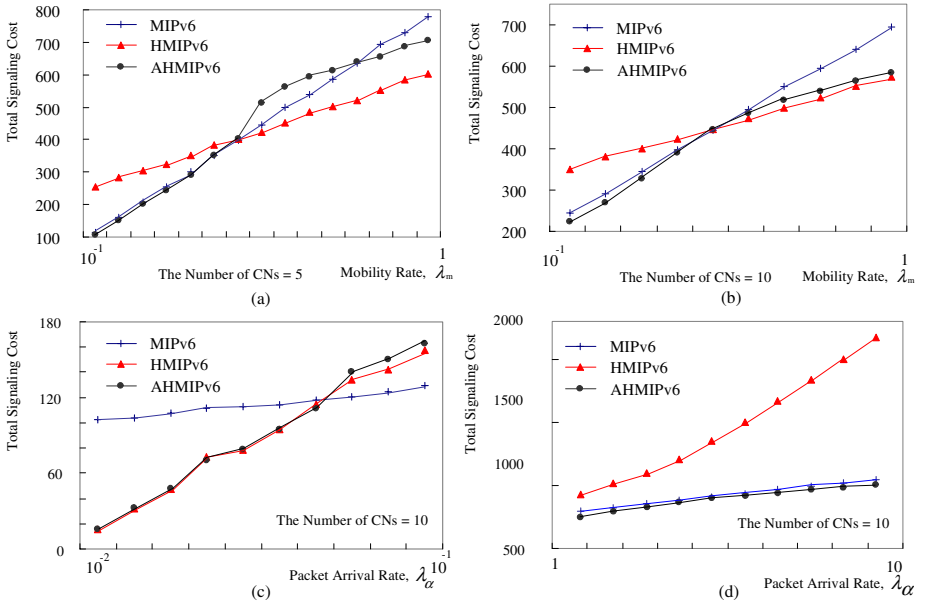


Fig. 3. Effects of Mobility Rate and λ_α on the Total Signaling Cost

results, the AHMIPv6 scheme has a considerable performance advantages over MIPv6 and HMIPv6. So, we conclude that the AHMIPv6 achieves significant performance improvements by using the MN's selection to send a BU message to the CN, either with an LCoA or an RCoA.

6 Conclusions

In this paper, we proposed an adaptive mobility management scheme for minimizing signaling costs in HMIPv6 networks. In our proposal, location registration with the HA and MAP is exactly the same as that in HMIPv6. However, the MN sends a BU message to the CN with either an LCoA or an RCoA, based on the geographical locality properties of the MN's movements. The cost analysis presented in this paper shows that the AHMIPv6 scheme achieves significant performance improvements by using the MN's selection to send a BU message to the CN, either with an LCoA or an RCoA.

References

1. Yi-Bing Lin, Shun-Ren Yang, "A Mobility Management Strategy for GPRS," *IEEE Transactions on Wireless Communications*, vol. 2 ,pp. 1178 - 1188, November 2003.
2. Akyildiz. I.F, Yi-Bing Lin, Wei-Ru Lai, Rong-Jaye Chen, "A New Random Walk Model for PCS Networks," *IEEE Journal on Selected Areas in Communications*, vol. 18, pp1254 - 1260, July 2000.

A Novel Tag Identification Algorithm for RFID System Using UHF

Ho-Seung Choi and Jae-Hyun Kim

School of Electrical and Computer Engineering, Ajou University,
San 5 Woncheon-Dong, Youngtong-Gu, Suwon 442-749, Korea
{lastjoin, jkim}@ajou.ac.kr

Abstract. An anti-collision algorithm is very important in the RFID system, because it decides tag identification time and tag identification accuracy. We propose improved anti-collision algorithms for RFID system using UHF. In the proposed algorithms, if the reader memorizes the Bin slot information, it can reduce the repetition of the unnecessary PingID command and the time to identify tags. If we also use ScrollAllID command in the proposed algorithm, the reader knows the sequence of collided ID bits. Using this sequence, we can reduce the repetition of PingID command and tag identification time. We analyze the performance of the proposed anti-collision algorithms and compare the performance of the proposed algorithms with that of the conventional algorithm. We also validate analytic results using simulation. According to the analysis, for the random tag ID, comparing the proposed algorithms with the conventional algorithm, the performance of the proposed algorithms is about 130% higher when the number of the tags is 200. For the sequential tag ID, the performance of the conventional algorithm decreases. On the contrary, the performance of the proposed algorithm using ScrollAllID command is about 16% higher than the case of using random tag ID.

1 Introduction

The RFID(Radio Frequency Identification) system is a simple form of ubiquitous sensor networks that are used to identify physical objects [1]. The RFID system identifies the unique tags' ID or detailed information saved in them attached to objects. Passive RFID systems generally consist of three components - a reader, passive tags, and a controller. A reader interrogates tags for their ID or detailed information through RF communication link, and contains internal storage, processing power, and so on. Tags get the processing power through RF communication link from the reader and use this energy to power any on-tag computations and to communicate to the reader. A reader in RFID system broadcasts the request message to the tags. Upon receiving the message, all the tags send the response back to the reader. If only one tag responds, the reader receives just one response. However, if two or more tags respond, their responses will collide on the RF communication channel, and thus cannot be received by the reader. The problem is referred to as the “Tag-collision”. An effective system must avoid this collision by using anti-collision algorithm be-

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 629 – 638, 2005.
© IFIP International Federation for Information Processing 2005

cause the ability to identify many tags simultaneously is crucial for many applications [2], [3]. Anti-collision algorithms are generally classified into ALOHA-based and binary-based methods. All these methods are based upon tags that are identified by a unique ID. The ALOHA-based anti-collision algorithms, which are probabilistic, are introduced in [4]-[7], and the anti-collision algorithms using binary-based method are introduced in [8]-[11]. There are two standard organizations of anti-collision in RFID system. One is ISO-18000 and the other is EPCglobal. In the resent RFID system, UHF(Ultra High Frequency) is more important issue than HF(High Frequency) since the reader can identify the tags faster and the tag can send more information using higher data rate. Therefore, we analyze the conventional anti-collision algorithm for EPC CLASS 1 RFID Tag operating in the frequency range of 860MHz-930MHz [12], which is named as EPC CLASS 1 UHF Tag. We also propose the improved anti-collision algorithms for EPC CLASS 1 UHF Tag. We mathematically compare the performance of the proposed algorithms with that of the conventional algorithm and also validate analytic results using OPNET simulation.

2 The Conventional Anti-collision Algorithm for EPC CLASS 1 UHF Tag

The anti-collision algorithm for EPC CLASS 1 UHF Tag resolves the collision by using PingID command and Bin slots which are used to receive tags' reply. The reply period for PingID command consists of 8 Bin slots. Each slot sequentially represents from '000' to '111'. The procedure of the algorithm using PingID command is as follows. First of all the reader transmits PingID command to the tags. The tags matching [VALUE] beginning at location [PTR] reply by sending 8-bits of the tag identifier beginning with the bit at location [PTR] + [LEN], where [VALUE] is the data that the tag will attempt to match against its own identifier(From the [PTR] position towards the LSB(Least Significant Bit)), [PTR] is a pointer to a location (or bit index) in the tag identifier, and [LEN] is the length of the data being sent in the [VALUE] field. The 8-bit reply is communicated during one of eight Bin slots delineated by the reader. The communication Bin slot is chosen to be equal to the value of the first 3 MSB(Most Significant Bit)s of the 8-bit reply. So, the tags whose 3 MSBs of ID after [VALUE] field is '000' choose Bin 0 and whose 3 MSBs of ID after [VALUE] field is '111' choose Bin 7. The reader sequentially processes Bin slots from Bin 0 to Bin 7. When two or more tags choose the same Bin slot, the reader retransmits PingID command to the tags. If only one tag chooses one Bin slot, the reader sends ScrollID command to the tag. The tags matching [VALUE] beginning at location [PTR] reply by sending their all ID. In this case only one tag sends its all ID to the reader and is identified by the reader. After identifying the first tag, the reader repeats same procedure using PingID command and ScrollID command until the reader identifies all the tags. Fig. 1 shows an example of the tag identification procedure using PingID and ScrollID commands. For more details, refer to [12].

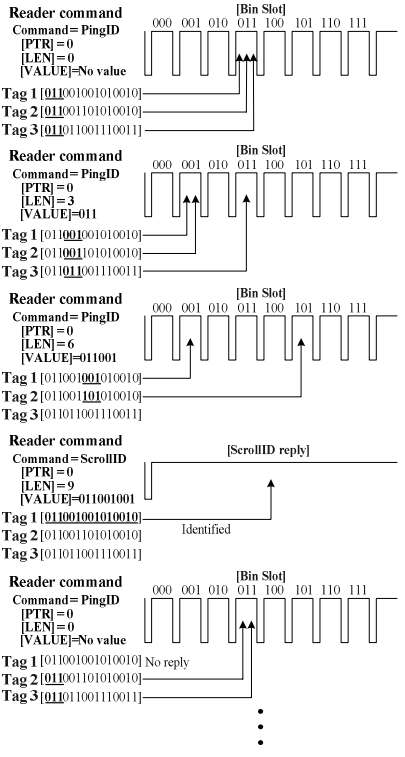


Fig. 1. An example of the conventional anti-collision algorithm

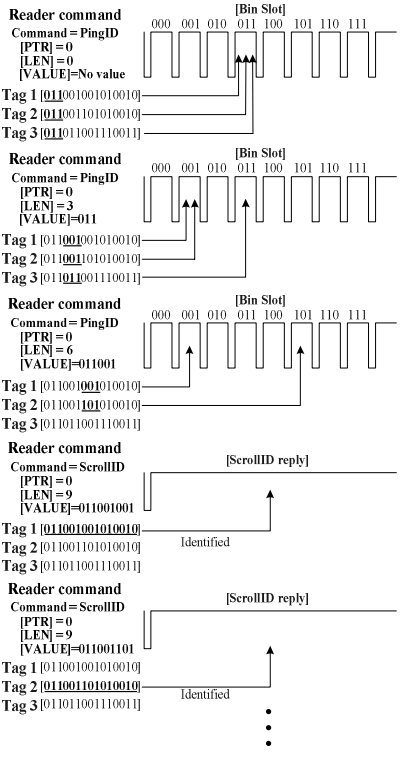


Fig. 2. An example of the proposed algorithm

3 The Proposed Anti-collision Algorithms

In the conventional anti-collision algorithm for EPC CLASS 1 UHF Tag, after the reader identifies a tag using PingID command and ScrollID command, the reader repeats all procedure to identify all the tags so that it takes much time to identify all the tags. If the reader uses the Bin slot information, it can reduce the repetition of unnecessary PingID command transmission and the time to identify tags. Fig. 2 shows an example of the tag identification procedure using the proposed anti-collision algorithm. In Fig. 1, after the reader identifies first tag, the reader goes back to the first procedure. In Fig. 2, the reader memorizes Bin slot information and using this information it can identify the second tag directly. In the conventional algorithm, the reader uses only PingID command and ScrollID command. In the case of using sequential tag ID, the conventional algorithm repeats PingID command much more times than the case of using random tag ID then it results in the longer tag identification time. If we use ScrollAllID command defined in [12] in the proposed algorithm, the reader knows the sequence of collided ID bits. Using this sequence, the reader can reduce the repetition of PingID

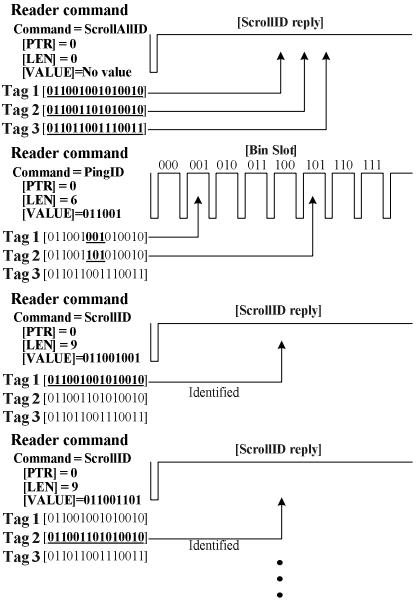


Fig. 3. An example of the proposed algorithm using ScrollAllID command

command and tag identification time. The procedure is as follows. First of all, the reader transmits ScrollAllID command to the tags. All the tags which are in active state send all of their ID bits to the reader. The reader knows the sequence of collided ID bits. So, the reader sets the [PTR] field to the position of second collided bit - 1 and transmits PingID command to the tags. Then, the tags whose first collided bit is '0' reply reader's PingID command, and the reader repeats transmission of PingID command until only one tag reply in one Bin slot. The reader transmits ScrollID command and identifies that tag. After the reader identifies the tag, the reader performs the next procedure using Bin slot information until all the tags are identified. The reader can reduce unnecessary repetition of PingID command. Fig. 3 shows an example of the tag identification procedure of the proposed anti-collision algorithm using ScrollAllID command.

4 Performance Analysis

In this section, we compare the performance of the conventional anti-collision algorithm with that of the proposed anti-collision algorithm for EPC CLASS 1 UHF Tag. For performance matrix, we consider the number of repetition of PingID command and the time to identify the tags.

4.1 The Conventional Anti-collision Algorithm for EPC CLASS 1 UHF Tag

We assume that tag ID is random, the number of total tags is m, and the number of Bin slot is *r*. Let q_n be the probability that there is no tag reply from Bin 0 to Bin *n*-1.

q_n is then $[(r\text{-}n)/r)]^m$. The probability that there is no reply in Bin n out of r-n slots is $[(r\text{-}1\text{-}n)/(r\text{-}n)]^m$, and the probability that there is only one reply in Bin n out of r-n slots is $[m/(r\text{-}n)]\times[(r\text{-}1\text{-}n)/(r\text{-}n)]^{m\text{-}1}$. The probability that there are two or more tag replies in Bin n out of r-n slots, q_{rn}, can be derived in

$$q_{rn}=\left[1-\left(\frac{r-1-n}{r-n}\right)^m-m\left(\frac{r-1-n}{r-n}\right)^{m-1}\cdot\frac{1}{r-n}\right]. \tag{1}$$

Let p_{1n} be the probability that there is no tag reply from Bin 0 to Bin n-1 and there are two or more tag replies in Bin n when the reader transmits first PingID command to the tags. Then, p_{1n} is calculated by

$$p_{1n}=\left(\frac{r-n}{r}\right)^m\left[1-\left(\frac{r-1-n}{r-n}\right)^m-m\left(\frac{r-1-n}{r-n}\right)^{m-1}\cdot\frac{1}{r-n}\right] \tag{2}$$

where r is 8 and $0\le n\le 7$ (n is an integer). Let p_1 be the probability that the reader retransmits PingID command in the first state. p_1 is then derived in

$$p_1=\sum_{n=0}^{7}p_{1n}\,. \tag{3}$$

We define that p_{2n} is the probability that there is no tag reply from Bin 0 to Bin n-1 and there are two or more tags' replies in Bin n when the reader transmits second PingID command to the tags. Since the number of Bin slot is r, we can approximate the number of tags in the second state by m/r. Then, m can be substituted by m/r in the second state. p_{2n} is expressed as

$$p_{2n}=\left(\frac{r-n}{r}\right)^{\frac{m}{r}}\left[1-\left(\frac{r-1-n}{r-n}\right)^{\frac{m}{r}}-\frac{m}{r}\left(\frac{r-1-n}{r-n}\right)^{\frac{m}{r}-1}\cdot\frac{1}{r-n}\right]. \tag{4}$$

The probability p_2 that the reader re-transmits PingID command in the second state is derived in

$$p_2=\sum_{n=0}^{7}p_{2n}\,. \tag{5}$$

Using same method, we can also approximate the number of tags in the k-th state by $m/r^{k\text{-}1}$. The probability p_k that the reader re-transmits PingID command in the k-th state is

$$p_k=\sum_{n=0}^{7}p_{kn}=\sum_{n=0}^{7}\left(\frac{r-n}{r}\right)^{\frac{m}{r^{k-1}}}\left[1-\left(\frac{r-1-n}{r-n}\right)^{\frac{m}{r^{k-1}}}-\frac{m}{r^{k-1}}\left(\frac{r-1-n}{r-n}\right)^{\frac{m}{r^{k-1}}-1}\cdot\frac{1}{r-n}\right]. \tag{6}$$

Therefore, for the first identified tag, the number of repetition of PingID command, I_1, can be calculated by

$$I_1 = 1 + \sum_{k=0}^{\infty} \sum_{n=0}^{7} \left(\frac{r-n}{r} \right)^{\frac{m}{r^k}} \left[1 - \left(\frac{r-1-n}{r-n} \right)^{\frac{m}{r^k}} - \frac{m}{r^k} \left(\frac{r-1-n}{r-n} \right)^{\frac{m}{r^k}-1} \cdot \frac{1}{r-n} \right], \quad \frac{m}{r^k} > 1 . \tag{7}$$

After the reader identifies a tag, the number of remaining tags is *m*-1. The reader repeats the same procedure to identify second tag. Using (7), the number of repetition of total PingID command until the reader identifies all the tags, I_{total}, is given by

$$I_{total} = m + \sum_{L=2}^{m} \sum_{k=0}^{\infty} \sum_{n=0}^{7} \left(\frac{r-n}{r} \right)^{\frac{L}{r^k}} \left[1 - \left(\frac{r-1-n}{r-n} \right)^{\frac{L}{r^k}} - \frac{L}{r^k} \left(\frac{r-1-n}{r-n} \right)^{\frac{L}{r^k}-1} \cdot \frac{1}{r-n} \right], \quad \frac{m}{r^k} > 1 . \tag{8}$$

The reader repeats the transmission of ScrollID command *m* times to identify *m* tags. The total time to identify *m* tags is composed of ScrollID command transmission time, PingID command transmission time, tag response time, transmission delay of the reader, and transmission delay of the tag. We assume that the size of the packet from reader to tag is *RL_C* and the data rate of the reader is DR_{reader}. Then, command packet transmission time from reader to tag to identify *m* tags, t_{reader}, is derived in

$$t_{reader} = \frac{RL_C(I_{total} + m)}{DR_{reader}} \tag{9}$$

where *RL_C* is 147 bits and DR_{reader} is 70,180 bps [12].

Assuming that the size of the tag response packet for PingID command is *TL_P*, the size of the tag response packet for ScrollID command is *TL_S*, and the data rate of the tag is DR_{tag}. Then, the response time of the tag to identify *m* tags, t_{tag}, is

$$t_{tag} = \frac{TL_P \times I_{total} + TL_S \times m}{DR_{tag}} \tag{10}$$

where *TL_P* is 8 bits, *TL_S* is 120 bits, and DR_{tag} is 140,350 bps [12].

To find the total delay to identify *m* tags, t_{delay} , we define that transmission delay of the reader is DE_{reader} and transmission delay of the tag is DE_{tag} , and t_{delay} is

$$t_{delay} = \left(I_{total} + m \right) \left(DE_{reader} + DE_{tag} \right) \tag{11}$$

where DE_{reader} is 17.81μs and DE_{tag} is 57μs [12]. Finally, we find the total time to identify *m* tags t_{total} in

$$t_{total} = t_{reader} + t_{tag} + t_{delay} . \tag{12}$$

4.2 The Proposed Anti-collision Algorithms

Using the same assumption in the section 4.1, the number of total tags is *m*, and the number of Bin slot is *r*. Tag ID is also random. Let p_{col} be the probability that there are two or more tag's replies in one Bin slot when the reader transmits the first PingID command to the tags. Then, p_{col} can be calculated as follows

$$p_{col} = \left[1 - \left(\frac{r-1}{r}\right)^{m} - m\left(\frac{r-1}{r}\right)^{m-1} \cdot \frac{1}{r}\right]. \tag{13}$$

Because the number of Bin slot is r, the number of repetition of PingID command in the first state, I_1, is given by

$$I_1 = r \times \left[1 - \left(\frac{r-1}{r}\right)^{m} - m\left(\frac{r-1}{r}\right)^{m-1} \cdot \frac{1}{r}\right]. \tag{14}$$

In the case of the proposed algorithm, the reader can memorize Bin slot information so that the number of repetition of PingID command in the k-th state, I_k, (k=1,2,3, …) is

$$I_k = r^k \times \left[1 - \left(\frac{r-1}{r}\right)^{\frac{m}{k-1}} - \frac{m}{r^{k-1}}\left(\frac{r-1}{r}\right)^{\frac{m}{k-1}-1} \cdot \frac{1}{r}\right]. \tag{15}$$

Finally, the number of repetition of total PingID command until the reader identifies all the tags, I_{total}, can be represented as

$$I_{total} = \sum_{k=0}^{\infty} I_k = I_0 + \sum_{k=1}^{\infty} r^k \times \left[1 - \left(\frac{r-1}{r}\right)^{\frac{m}{r^{k-1}}} - \frac{m}{r^{k-1}}\left(\frac{r-1}{r}\right)^{\frac{m}{r^{k-1}}-1} \cdot \frac{1}{r}\right], \quad \frac{m}{r^{k-1}} > 1. \tag{16}$$

I_0 is the number of the first transmission of PingID command and the value of I_0 is 1.

In the proposed algorithms, we can calculate total time to identify m tags, t_{total}, by (12). In the case of using ScrollAllID command, we can calculate total time to identify m tags, t_{total}, as same as in (12) with substituting m by m+1.

5 Analytic and Simulation Results

In this section, we mathematically compare the performance of the proposed algorithm with that of the conventional anti-collision algorithm for EPC CLASS 1 UHF Tag and validate analytic results using simulation. Tag's ID is 96 bits and the portion of the ID to identify the tags is 36 bits. To maximize the reliability of simulation, we apply the real packet size, packet transmission delay, and date rate [12].

Fig. 4 shows the number of PingID command for the number of used tags in the conventional anti- collision algorithm for EPC CLASS 1 UHF Tag, the proposed algorithm not using ScrollAllID command, and the proposed algorithm using ScrollAllID command for random tag ID. Lines and symbols represent analytic and simulation results, respectively. In Fig. 4, we observe the small difference between analytic and simulation results in the conventional algorithm. In the analytic result, we do not consider the addictive number of PingID command in the condition of $m/r^k \le 1$. That is why there is a small difference between analytic and simulation results. In Fig. 4, the number of PingID command of the proposed algorithms is less than that of the conventional algorithm.

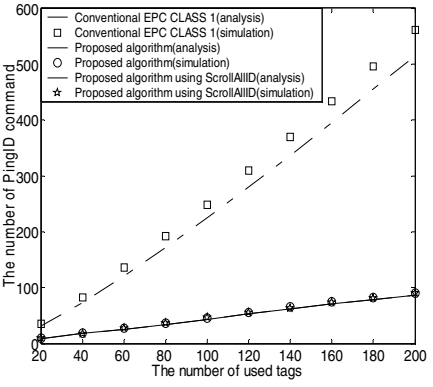


Fig. 4. The number of PingID command for the number of used tags(random tag ID)

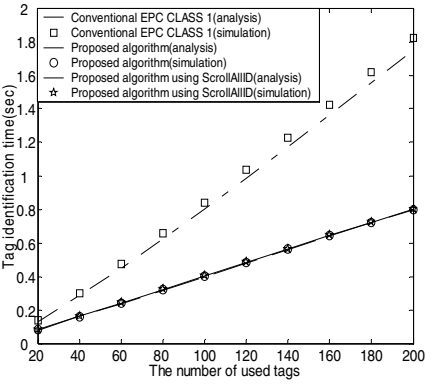


Fig. 5. Tag identification time for the number of used tags(random tag ID)

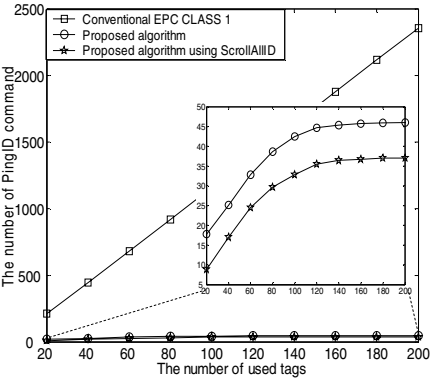


Fig. 6. The number of PingID command for the number of used tags(sequential tag ID)

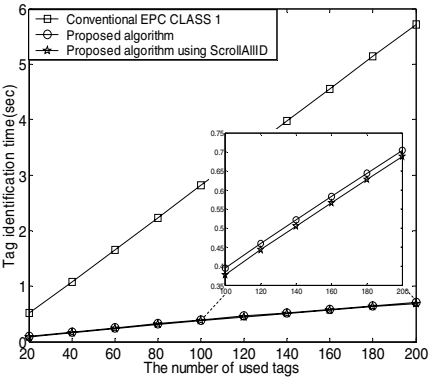


Fig. 7. Tag identification time for the number of used tags(sequential tag ID)

Fig. 5 presents the tag identification time for the number of used tags of each algorithm for random tag ID. Lines and symbols represent analytic and simulation results, respectively. In Fig. 5, the tag identification time for the same number of used tags of the proposed algorithms is much less than that of the conventional algorithm. The performance of the proposed algorithms is about 70% higher than that of the conventional algorithm for the 20 tags and about 130% higher for the 200 tags. Tag identification rate of the conventional algorithm is 117 tags/sec and that of the proposed algorithms is 252 tags/sec.

In Fig. 6, we show the number of PingID command for the number of used tags of each algorithm for the sequential tag ID. Fig. 6 represents the simulation result. In this figure, in the conventional algorithm for the sequential tag ID, the number of PingID command is very large. In the case of the conventional algorithm, when the reader transmits PingID command, many tags reply in the same Bin slot. That is why the number of PingID command is larger. On the contrary, the number of PingID com-

mand of the proposed algorithms is very smaller than the case of using random tag ID. When the reader uses ScrollAllID command, it knows the sequence of the collided bits so that unnecessary transmission of PingID command can be reduced.

Fig. 7 illustrates the tag identification time for the number of used tags of each algorithm. This result is for the sequential tag ID and represents the simulation result. In Fig. 7, the tag identification time of the conventional algorithm increases comparing with the case of using random tag ID. However, for the 200 tags, we found that the performance improvement of the proposed algorithm not using ScrollAllID command is about 13% and that of the proposed algorithm using ScrollAllID command is about 16%, respectively.

6 Conclusion

We analyzed the conventional anti-collision algorithm for EPC CLASS 1 UHF Tag and proposed the improved anti-collision algorithms. In the proposed algorithms, if the reader memorizes the Bin slot information, it can reduce the repetition of unnecessary PingID command and the time to identify tags. If we also use ScrollAllID command in the proposed algorithm, the reader knows the sequence of collided ID bits. Using this sequence, the reader can reduce the repetition of PingID command and tag identification time. We compared the performance of the proposed algorithms with that of the conventional anti-collision algorithm for the random tag ID and sequential tag ID, respectively. We also validated analytic results using simulation. In the case of using random tag ID, the performance of the proposed algorithms is about 130% higher for the 200 tags. Tag identification rate of the conventional algorithm is 117 tags/sec and that of the proposed algorithms is 252 tags/sec. For many practical RFID applications using sequential tag ID, the conventional anti-collision algorithm shows performance degradation since the reader transmits many PingID commands. However, the proposed algorithm using ScrollAllID command shows performance improvement as much as about 16%. In conclusion, the proposed algorithms will contribute to improve the performance of the RFID system because the reader can identify more tags with shorter time.

Acknowledgment. This research is partially supported by the Ubiquitous Autonomic Computing and Network Project, the Ministry of Science and Technology(MOST) 21st Century Frontier R&D Program in Korea.

References

1. S. Sarma, D. Brock, and D.Engels, “Radio frequency identification and electronic product code,” 2001 IEEE MICRO, 2001.
2. F. Zhou, D. Jin, C. Huang, and M. Hao, “Optimize the power consumption of Electronic Passive Tags for Anti-collision Schemes,” IEEE, 2003.
3. S. Sarma, J. Waldrop, and D. Engels, "Colorwave : An Anti-collision Algorithm for the Reader Collision Problem," IEEE International Conference on Communications, ICC '03, vol. 2, May 2003, pp. 1206-1210.

4. H. Vogt, "Efficient Object Identification with Passive RFID tags," International Conference on Pervasive Computing, pp.98-113, Zürich, 2002.
5. C. S. Kim, K. L. Park, H. C. Kim and S. D. Kim, "An Efficient Stochastic Anti-collision Algorithm using Bit-Slot Mechanism," PDPTA04, 2004.
6. S. R. Lee, S. D. Joo, C. W. Lee, "High-Speed Access Technology of Tag Identification Using Advanced Framed Slotted ALOHA in an RFID System," Journal of IEEK, Vol. 41-TC, NO.9, Sep., 2004, pp. 29-37.
7. ISO/IEC 18000-6:2003(E), Part 6: *Parameters for air interface communications at 860-960 MHz*, Nov. 26, 2003.
8. J. L. Massey, "Collision resolution algorithms and random-access communications," Univ. California, Los Angeles, Tech. Rep. UCLAENG -8016, Apr. 1980.
9. K. Finkenzeller, RFID Handbook: *Radio-Frequency Identification Fundamentals and applications*. John Wiley & Sons Ltd, 1999.
10. H. S. Choi, J. R. Cha and J. H. Kim, "Fast Wireless Anti-collision Algorithm in Ubiquitous ID System ," in *Proc. IEEE VTC 2004*, L.A., USA, Sep. 26-29, 2004.
11. H. S. Choi, J. R. Cha and J. H. Kim, "Improved Bit-by-bit Binary Tree Algorithm in Ubiquitous ID System," in *Proc. IEEE PCM 2004*, Tokyo, Japan, Nov. 29-Dec. 3, 2004, pp. 696-703.
12. EPCglobal, *EPC^{TM} Tag Data Standards Version 1.1 Rev.1.24*, Apr. 2004.

Coverage-Aware Sensor Engagement in Dense Sensor Networks*

Jun Lu, Lichun Bao, and Tatsuya Suda

Bren School of Information and Computer Sciences,
University of California, Irvine, CA 92697
{lujun, lbao, suda}@ics.uci.edu

Abstract. The critical issue in sensor networks is the search of balance between the limited battery supply and the expected longevity of network operations. Similar goals exist in providing a certain degree of sensing coverage and maintaining a desirable number of sensors to communicate under the energy constraint. We propose a novel sensor network protocol, called Coverage-Aware Sensor Engagement (CASE) for coverage maintenance. Different from others, CASE schedules active/inactive sensing states of a sensor according to the sensor's contribution to the network sensing coverage. The contribution is quantitatively measured by a metric called coverage merit. By utilizing sensors with large coverage merit, CASE reduces the number of the active sensors required to maintain the level of coverage. Simulation results show that CASE considerably improves the energy efficiency and reduces the computation and communication costs to maintain the required coverage degree in a dense sensor network.

1 Introduction

Wireless sensor networks are networks of a large number of small wireless devices, which collaborate to monitor environments and report sensing data via wireless channels. Wireless sensor networks have emerged rapidly in a variety of applications. For instance, thermal sensors are being deployed to monitor temperature in a forest, and to report the temperature information back to data collection nodes for further analysis. In another instance, a large number of seismic sensors are employed to monitor animal activities in a wild field. The seismic sensors, when triggered by the vibrations caused by animal movements, can record the vibration signals and report them to data collection nodes. Information about animal activities, like their moving tracks and velocities, can be acquired through analyzing the collected signals.

Wireless sensors are very limited in their processing, computing and communication capabilities as well as the storage and power supply. The typical Crossbow MICA mote MPR300CB [XBOW1] has a low-speed 4MHz processor equipped with only 128KB

* This work was supported by the National Science Foundation through grants ANI-0083074 and ANI-9903427, by DARPA through Grant MDA972-99-1-0007, by AFOSR through Grant MURI F49620-00-1-0330, and by grants from the University of California MICRO Program, Hitachi, Hitachi America, Novell, Nippon Telegraph and Telephone Corporation (NTT), NTT Docomo, Fujitsu, and NS-Solutions.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 639–650, 2005.
© IFIP International Federation for Information Processing 2005

flash, 4KB SRAM and 4KB EEPROM. It has a maximal data rate of 40kbps and a transmission range of about 100 feet, powered by two AA batteries. Therefore, a wireless sensor network is usually deployed with high density. Dense deployment not only helps to improve a sensor network's reliability, but also extends its longevity. In practice, large-scale wireless sensor networks are usually deployed randomly.

Given such a randomly and densely deployed wireless sensor network, it is desirable to have sensors autonomously schedule their duty cycles while satisfying the degree of sensing coverage required by the application. The problem is called coverage maintenance.

Coverage maintenance problem in sensor networks has drawn intense research attention recently. Tian *et al.* [TG1] presented a node-scheduling algorithm to turn off redundant sensors if their sensing areas are already covered by their neighbors. Randomized as well as coordinated sleep algorithms were proposed in [HL1] to maintain network coverage using low duty-cycle sensors. The randomized algorithm enables each sensor to independently sleep under a certain probability. The coordinated sleep algorithm allows each sensor to enter sleep state if its sensing area is fully contained by the union set of its neighbors. A K-coverage maintenance algorithm was proposed in [HT1] so that each location of the sensing area is covered by at least K sensors. A sensor decides whether it is redundant only by checking the coverage state of its sensing perimeter. In [GWL1], the redundancy of the sensing coverage of wireless sensor networks is analyzed, and the relation between the number of neighbors and the coverage redundancy is studied. Abrams *et al.* studied a variant of the NP-hard SET K-COVER problem in [AGP1], partitioning the sensors into K covers such that as many areas are monitored as frequently as possible. Yan *et al.* proposed an adaptable energy-efficient sensing coverage protocol, in which each sensor broadcasts a random time reference point, and decides its duty schedule based on neighbors' time reference points [YHS1].

We propose a new coverage maintenance scheme called Coverage-Aware Sensor Engagement (CASE). CASE is based on a probabilistic sensing model, which is more practical than the disk sensing model assumed by many others. Rather than fixing the sensing range of a sensor as the disk sensing model, the probabilistic sensing model defines the sensing ability as the probability to detect an event happening at a location. To the best of our knowledge, this work is the first to address the K-coverage problem under the probabilistic sensing model. In fact, the disk sensing model is a special case of the probabilistic sensing model, and CASE works for the disk sensing model as well. In CASE, each sensor is initially inactive in sensing, but checks to see whether it is necessary to turn on its sensing unit according to its contribution (which we called coverage merit) to meet the required degree of sensing coverage. Before actually turning on itself, each sensor waits for a back-off period decided by its coverage merit. Sensors with larger coverage merit have shorter back-off period. In this way, sensors turn on themselves (if necessary) in the decreasing order of their coverage merit. By utilizing sensors with large coverage merit, CASE can reduce the active sensor density needed to maintain the required coverage degree.

The rest of this paper is organized as follows. The differences of our work from the others are examined in Section 2. Section 3 describes the assumptions of CASE.

Section 4 specifies CASE in more details. Simulation results are presented in Section 5 for performance evaluations. Section 6 concludes the paper.

2 Prior Works

In the scheduling algorithm proposed by [TG1], every sensor is active at the beginning. A sensor is eligible to turn off if its sensing area is covered by the union of the sponsored sectors by its neighbors. A sensor eligible to turn off broadcasts a `TURNOFF` beacon to inform neighbors. Every other sensor receiving such a beacon re-evaluates its eligibility to turn off. If not eligible, the sensor cancels the timer and stays active. [YHS1] presented an elegant approach to dynamically schedule sensors in order to guarantee a certain degree of coverage. Each sensor generates a random reference time and exchanges the reference time with its neighbors. Each sensor can setup its working schedule by examining the reference time of its neighbors.

There are several major differences between the proposed algorithm and the algorithms proposed in [TG1] and [YHS1]. First, we differentiate sensors according to their coverage merit, which decreases the active sensor density to provide the coverage degree required by the application. Second, while both of the existing schemes assume the disk sensing model, CASE solves the K-coverage problem under the probabilistic sensing model. Treating the disk sensing model as a special case of the probabilistic model, CASE also works for the disk sensing model. Third, unlike [TG1], CASE sets sensor initial state as inactive. Following the scheduling procedure, each sensor tries to turn on to provide the required coverage degree. This feature is favorable for dense deployment in that the communication and computation overhead is reduced due to less sensor state changes.

The scheme proposed by [TG1] does not work for the probabilistic sensing model. In order to compare CASE with the scheme proposed in [TG1], we modified the eligibility rule of [TG1] to accommodate the probabilistic sensing model. We refer the scheme proposed in [TG1] as *the sponsored sector scheduling scheme* or *Tian-Sector* and the scheme with the modified eligibility rule as *the grid point scheduling scheme* or *Tian-Grid*. Section 5.1 explains in details about the differences between Tian-Grid and Tian-Sector. The validation of the working schedule setup algorithm in [YHS1] is tightly coupled with the assumption of the disk sensing model, which prevents it from being ported to the probabilistic sensing model. Therefore, in the simulation evaluation, we only compare CASE with Tian-Grid and Tian-Sector.

3 Assumptions

We assume that sensors are static, and that each sensor knows its location as well as its neighbors'. Such assumptions are conveniently taken by other works [HT1] [TG1] [YHS1] and are supported by the existing research [ACZ1] [BHE1] [BP1] [PCB1]. The location information can be absolute or relative to neighbors. We also assume that sensors can synchronize their timers [DH1] [EGE1].

We assume that the sensing ability model of sensors is available before deployment through calibration process. A sensor detects an event based on its measurement, and

the event is detected if the measurement strength is above a preset threshold. Due to the signal attenuation and noise, a sensor's measurement is modeled by a probability density function (PDF), which varies with the type of signals and the propagation channel. In CASE, the sensing ability of a sensor is modeled as the probability of a successful detection of certain events of interests.

Apparently, a sensor's sensing ability is a function of the distance between the sensor and the event [MKQP1]. We use $S_j(P_i)$ to describe sensor j's sensing ability at location P_i. A sensor's sensing range, which is denoted by SR, is defined as the range, beyond which the sensor's sensing ability can be neglected. The disk sensing model is regarded as a special case of the probabilistic sensing model, where a sensor detects an event within the SR with the probability 1 and outside the SR with the probability 0.

We assume that sensors have the same SR, and that the sensor communication range is greater than or equal to $2{\cdot}SR$. This is usually true in practice. For example, ultra-sonic sensors have a sensing range of approximately 0.2-6m [ROB1] while the transmission range of MICA motes is about 30 meters [XBOW1]. In the cases that the communication range is less than $2 \cdot SR$, our algorithms can work through multi-hop transmissions.

4 Coverage-Aware Sensor Engagement (CASE)

4.1 *K*-Coverage

The objective of CASE is to guarantee K-Coverage with the least number of active sensors. Under the disk sensing model, location P_i is K-covered if the location is monitored by at least K sensors. Under the probabilistic sensing model, however, we need to modify the definition of K-Coverage. We say a location P_i is K-covered if the *expectation* of the number of sensors that monitor an event at the location is at least K, or essentially the weighted sum of active sensors is no less than K, as shown in Eq. (1).

$$\sum_j S_j(P_i) \geq K \tag{1}$$

where $S_j(P_i)$ is the probability of sensor j to detect an event at location P_i. Note that coverage degree K can be a real number under the probabilistic sensing model. For example, an application may require the target area to be 1.5-covered, which means the expected number of sensors that detect an event at any location in the area needs to be at least 1.5.

4.2 Coverage Merit

Apparently, when a location P_i is already covered by a group of sensors A, the additional coverage needed to fulfill the K-coverage requirement is

$$C(P_i) = K - \sum_{m \in A} S_m(P_i). \tag{2}$$

If $C(P_i)$ is greater than 0, more sensors are required to provide additional coverage.

In order to see sensor j's coverage merit at location P_i, we compare sensor j's probability of detecting an event at location P_i with $C(P_i)$, the minimum of which is defined as sensor j's coverage merit at the location. That is, sensor j's coverage merit at location P_i is:

$$CM_j(P_i) = \begin{cases} \min(C(P_i), S_j(P_i)), & C(P_i) > 0 \\ 0, & C(P_i) \leq 0 \end{cases} \quad (3)$$

Note that when $C(P_i)$ is less than or equal to 0, P_i is already K-covered, therefore the sensor j's coverage merit at this location is 0. It is easy to see that $CM_j(P_i)$ is a continuous function over the sensing area of sensor j, and is dependent on the active states of its neighbors.

In order to evaluate sensor j's coverage contribution to the sensor network as a whole, the summation of $CM_j(P_i)$ over sensor j's sensing area is computed as:

$$CM_j = \int\int CM_j(P_i)dxdy$$

Since the existence of a sensor only affects the area covered by the sensor, its coverage merit can be calculated by only considering the area within its SR. For computation convenience, the above equation is converted into polar coordinates:

$$CM_j = \int_0^{2\pi} \int_0^{SR} CM_j(P_i)rd\theta dr \quad (4)$$

4.3 Coverage-Aware Sensor Engagement

To provide K-coverage with the minimum number of sensors, CASE applies a greedy strategy by gradually activating sensors in decreasing order of their coverage merit. In contrast, previous schemes schedule sensors regardless of their contribution to meet the required degree of network coverage (*e.g.*, in [TG1], redundant sensors have the same chance to power off based on a random back-off timer). More specifically, CASE runs in two phases as follows:

1. Wakeup phase: the first phase when sensors start in inactive sensing state, and gradually enter the active state according to their coverage merits.
2. Optimization phase: the second phase when sensors optimize the coverage by turning off redundant sensors to meet coverage requirements.

In the wakeup phase, each sensor is inactive in sensing, and computes an initial coverage merit. Note that the inactive/active states are logical states in CASE, *i.e.*, sensors are actually active to execute the CASE algorithm. Because no neighbor is active, the initial coverage merit of a sensor is maximum, and given by Eq. (5).

$$CM_{max} = \int_0^{2\pi} \int_0^{SR} \min(K, S_j(P_i))rd\theta dr \quad (5)$$

Afterward, each sensor sets a back-off timer T to announce its active state. The back-off timer T is determined according to its coverage merit using Eq. (6).

$$T = \xi \cdot (CM_{max} - CM_j) + \epsilon \tag{6}$$

where ξ is a configurable system parameter, and ϵ is a small positive random number. ξ determines the convergence latency of the wakeup phase in CASE. Small value of ξ means fast convergence but may increase the chance of collisions among neighboring sensors. The choice of an appropriate value for ξ is out of the scope of this paper, and will be part of our future work. ϵ is used to prevent the potential collision between two neighboring sensors with the same coverage merit.

According to Eq. (6), sensors with larger coverage merit have shorter back-off period. When a sensor times out, the sensor changes to active state, and broadcasts a `TURNON` message to its neighbors within its transmission range, which is approximated by $2 \cdot SR$. When a sensor receives a `TURNON` message before the timer expires, it recalculates its coverage merit, and adjusts its back-off timer accordingly.

According to Eq. (2), (3) and (4), the sensor's coverage merit is reduced when a new neighbor is turned into active state, thus the back-off timer is always delayed. Once all the locations within the sensing range of a sensor are K-covered, the sensor's coverage merit becomes 0. The sensor cancels its back-off timer and stays inactive.

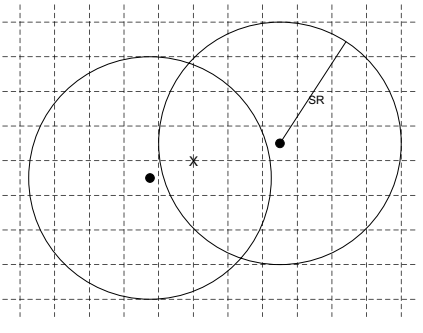


Fig. 1. Grid Point X in the Target Area

The wakeup phase ends at $\xi \cdot CM_{max}$. After the wakeup phase, there may be redundant active sensors. This is because that the coverage of the sensors turning on later may overlap with the sensing area of the active sensors. In the optimization phase, we use a similar random back-off algorithm as [TG1] to turn off redundant sensors. Accordingly, each redundant sensor sets a random timer, and re-checks its eligibility to turn off when received `TURNOFF` messages from other sensors. If the sensor realizes that it is not eligible to turn off, it cancels its timer and stays active. Otherwise, it broadcasts a `TURNOFF` message and turns off upon timeout.

In order to simplify the computation, we cover the target area by a virtual square grid (Fig. 1) and sensors only consider the grid points within the SR when calculating coverage merit (this technique is also used by Yan *et. al* [YHS1]). The coverage merit

is approximated by the summation of the coverage merit on the grid points within the SR, *i.e.*, P_i in Eq. (4) is a grid point.

5 Simulation Evaluations

5.1 Experiment Setup

We carried out experiments under two sensing models — the probabilistic sensing model and the disk sensing model, both over a square deployment area of $100 \times 100 m^2$.

In Eq. (6), parameters are chosen as $\xi = 0.1$ and $\epsilon = 0.01$. If not otherwise specified, the deployment density is set to 0.08 sensors/m^2, and the network is designed to provide the coverage degree of 1.0.

The probabilistic sensing models depend on the sensor capabilities and environments. Although CASE shall work with any realistic sensing model, for simplicity, we assume a virtual probabilistic sensing model for the sensors, two examples of which are shown below,

$$S_j(P_i) = f(D_{ij}) = \frac{1}{1 + \alpha D_{ij} + \beta D_{ij}^2 + \cdots + \gamma D_{ij}^k}$$

$$S_j(P_i) = f(D_{ij}) = \frac{1}{\chi^{D_{ij}}}$$

where D_{ij} is the distance between sensor j and location P_i; α, β, γ and χ ($\chi > 1$) are system parameters reflecting the physical characteristics of sensor j and deployment environments.

Specifically, we assume the following virtual probabilistic sensing model in the simulations:

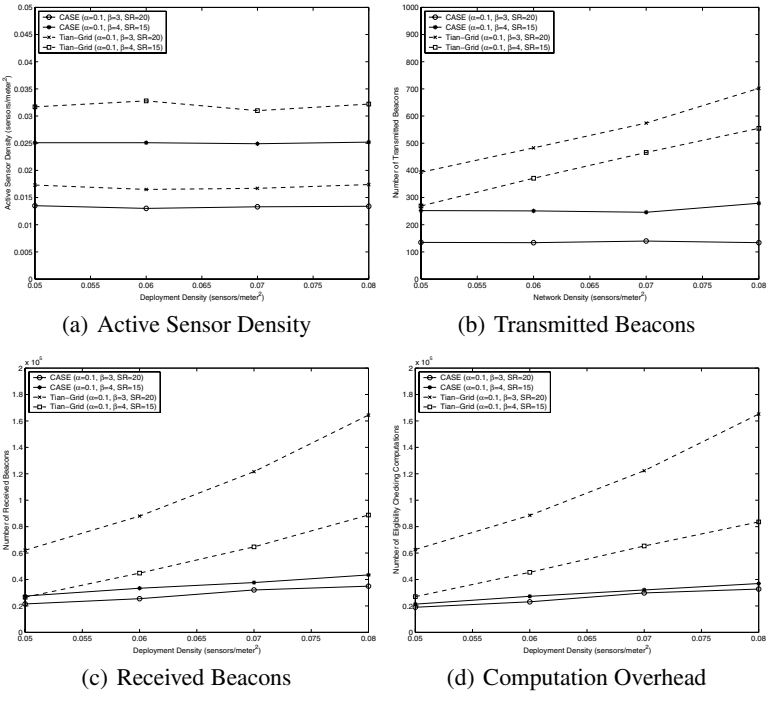
$$f(D_{ij}) = \frac{1}{(1 + \alpha D_{ij})^\beta} \tag{7}$$

where α is set to 0.1 and β is set to 3 or 4. Assuming that detection probability lower than 4% is negligible, two SRs, *i.e.*,15 and 20 meters, are simulated. For the disk sensing model, the SR is set to 15 meters.

As we explained in Section 2, under the probabilistic model, CASE is compared with the modified Tian-Sector based on virtual grids, which we call Tian-Grid. Like CASE, Tian-Grid checks the expected number of monitoring sensors at each grid point within its sensing range. A sensor is eligible to turn off if the expected number of monitoring sensors of each grid point within its sensing range is at least K. Also, different from Tian-Sector, which only examines the sectors sponsored by neighbors within SR, Tian-Grid considers all the neighbors within $2 \cdot SR$. In the disk sensing model, we compare CASE with both Tian-Grid and Tian-Sector because Tian-Grid natively works for the disk sensing model.

5.2 Result Analysis

The simulation results show the performance of CASE in terms of active sensor density, communication overhead and computation overhead. The communication overhead is



(a) Active Sensor Density (b) Transmitted Beacons

(c) Received Beacons (d) Computation Overhead

Fig. 2. Various Deployment Densities

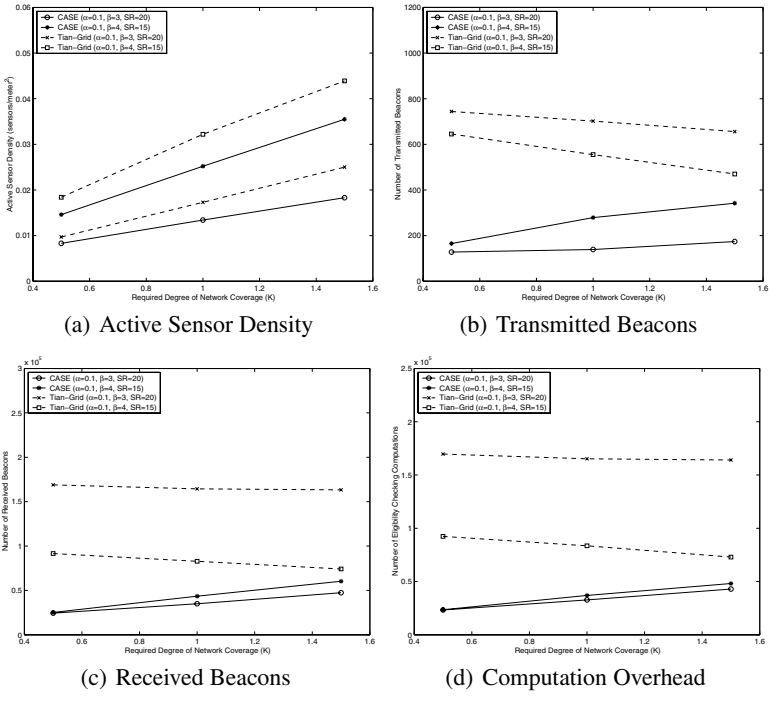
computed as the number of beacons sent and received for the `TURNON` messages in the wakeup phase and the `TURNOFF` messages in the optimization phase in CASE. Because the eligibility checking is the most costly computation operation, the computation overhead is calculated as the times of checking the eligibility of a sensor to be in active state, which is determined by coverage merit.

We analyze the results in the probabilistic sensing model and the disk sensing model, separately.

Probabilistic sensing model. In section 4.3, we have proposed to compute the coverage merit based on virtual grids. For comparison purposes, we simulate the modified Tian-Sector protocol, which we refer as *Tian-Grid* in the figures, and collect corresponding statistics.

The results under various deployment densities are shown in Fig. 2. Results for different required coverage degrees are shown in Fig. 3.

Fig. 2(a) indicates that both CASE and Tian-Grid provide stable active sensor density. However, CASE results lower active sensor density than Tian-Grid under different deployment densities because CASE activates sensors with large coverage merit, therefore allowing less active sensors in order to provide the same degree of coverage. For instance, when the sensor network has the deployment density of 0.05 sensors/m^2 and sensors have the SR of $20meters$, CASE provides 1.0-coverage with the active sensor density of only 0.0137 sensors/m^2, whereas Tian-Grid requires 0.0175 sensors/m^2.



(a) Active Sensor Density (b) Transmitted Beacons

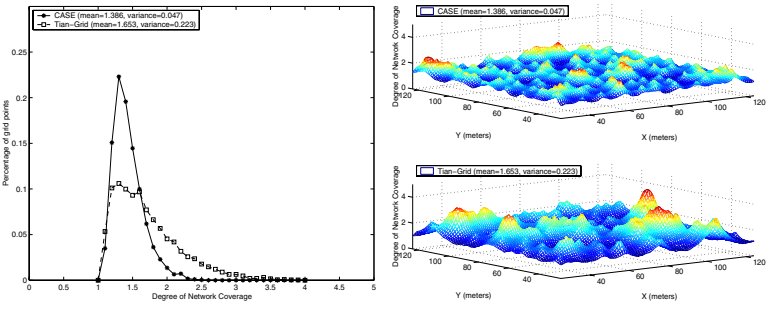
(c) Received Beacons (d) Computation Overhead

Fig. 3. Various Required Coverage Degrees (K)

Fig. 2(b) shows that CASE uses less beacons than Tian-Grid. This is due to the fact that sensors are gradually switched on from inactive state to active state in CASE, whereas Tian-Grid has all sensors initially in active state and turn off redundant sensors, which translates into different amount of beacons transmitted in order to inform state changes. If the network deployment is dense enough, the number of redundant sensors is much larger than the number of active sensors needed to provide the required coverage degree. Thus CASE involves less state changes than Tian-Grid.

Furthermore, we observe that the number of transmitted beacons in CASE changes little along with the increase of deployment density. In contrast, Tian-Grid suffers when the deployment density increases in Fig. 2(b). This is because the active sensor density is almost stable along with the deployment densities in CASE, whereas in Tian-Grid, most beacons are the `TURNOFF` messages sent by redundant sensors. When the deployment density increases, more redundant sensors need to turn off with more beacons.

Similar to Fig. 2(b), Fig. 2(c) shows that CASE has less received beacons than Tian-Grid, and that the number of beacons received in both schemes increases with the deployment density because of the broadcast nature of the wireless channel. However, the increasing rate of received beacons in CASE is less than that in Tian-Grid because the increase of the received beacons in CASE is mainly caused by the increase of sensor density. In Tian-Grid, however, the increase is caused by the increase of both the number of the transmitted beacons and sensor density. Because the eligibility checking



(a) (b)

Fig. 4. Coverage Distribution; $\alpha = 0.1, \beta = 3, SR = 20$

computations are often triggered by the received beacons, we have similar observation for computation overhead as shown in Fig. 2(d).

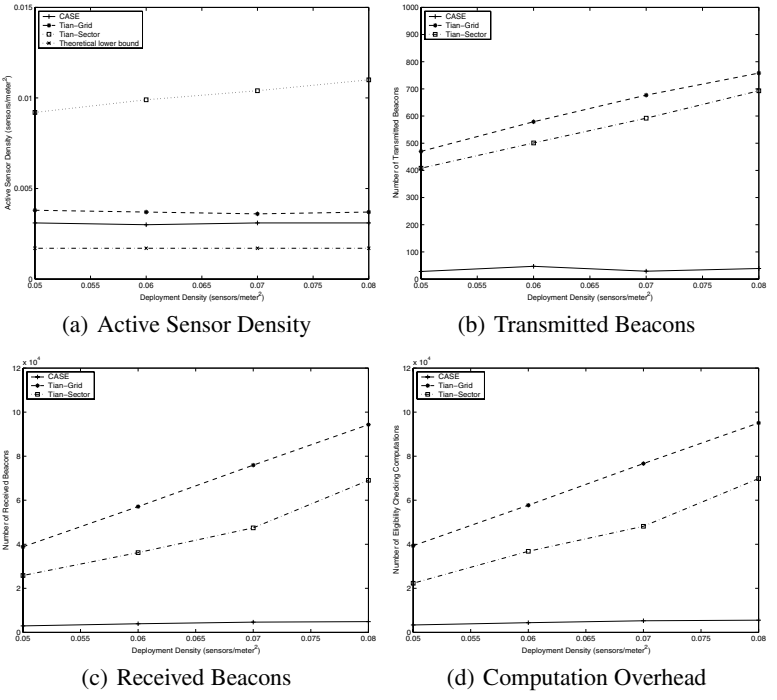
In Fig. 3, we show the results based on various coverage degrees requirements. Again, CASE performs better than Tian-Grid under various coverage degree requirements. However, the difference between the two protocols in Fig. 3(b), 3(c) and 3(d) diminishes with the increase of the required coverage degree. This is because the higher the coverage degree, the more sensors are needed active. Because the sensors initially assume inactive in CASE, higher coverage degree means more sensors need to turn on. While in Tian-Grid where sensors are initially active, higher coverage degree means less sensors need to turn off.

To further investigate the performance improvement of CASE, we show a normalized histogram of the number of grid points under different coverage degrees in Fig. 4(a). As we can see, majority of the grid points are covered by a degree from 1.0 to 2.0 in CASE, while the grid-point coverage under Tian-Grid varies from 1.0 to 3.0. From a different observation angle, we plotted the coverage degree of different points in the sensor network as shown in Fig. 4(b), which indicates that CASE provides more even coverage than Tian-Grid does.

Disk sensing model. We compare CASE with Tian-Grid and Tian-Sector under the disk sensing model in Fig. 5. In [YZLZ1], a theoretical lower bound on the active sensor density to achieve 1-coverage is provided as $2/\sqrt{27}SR^2$, and is again calculated here in Fig. 5(a) as a baseline for the comparison purposes. Note that although we only present the results of 1-coverage, similar results are observed for other K values (*e.g.*, $K = 2$).

Fig. 5(a) shows that Tian-Grid achieves the same required coverage degree with less than half of the active sensor density required by Tian-Sector. This is because that Tian-Sector is conservative about the sensor redundancy by only considering the neighbors within SR, and ignoring the coverage provided by sensors in range from SR to $2 \cdot SR$. Thus Tian-Sector results in relative high density of active sensors. Again, CASE performs better than Tian-Grid by reducing 20% of the active sensor density.

A larger discrepancy between CASE and the other two protocols are shown in terms of the communication and computation overheads in Fig. 5(b), 5(c) and 5(d).



(a) Active Sensor Density (b) Transmitted Beacons

(c) Received Beacons (d) Computation Overhead

Fig. 5. Disk Sensing Model

6 Conclusions

We have proposed a novel coverage maintenance scheme called Coverage-Aware Sensor Engagement (CASE). CASE conserves energy while providing the required coverage degree by allowing sensors to autonomously decide their active/inactive states. Unlike prior works, CASE considers local coverage information of sensors, *i.e.* coverage merit, when scheduling sensors' active/inactive states. Simulation results show that CASE provides the required coverage degree for a dense sensor network with lower active sensor density and less communication and computation costs than existing solutions. Furthermore, CASE is highly scalable to sensor network deployment density due to the low increasing rate of communication and computation costs relative to the increase of deployment density.

References

[ACZ1] Albowicz, J., Chen, A., Zhang, L.: Recursive Position Estimation in Sensor Networks. Proceedings the IEEE International Conference on Network Protocols (ICNP), 2001.

[AGP1] Abrams, Z., Goel, A., Plotkin, S.: Set K-Cover Algorithms for Energy Efficient Monitoring in Wireless Sensor Networks. Proceedings IPSN, 2004.

[BHE1] Bulusu, N., Heidemann, J., Estrin, D.: GPS-less Low Cost Outdoor Localization For Very Small Devices. IEEE Personal Communications, 2000.

[BP1] Bahl, P., Padmanabhan, V.: RADAR: An In-Building RF-based User Location and Tracking System. Proceedings INFOCOM, 2000.

[DH1] Dai, H., Han, R.: TSync: A Lightweight Bidirectional Time Synchronization Service for Wireless Sensor Networks. Mobile Computing and Communications Review, 2004.

[EGE1] Elson, J., Girod, L., Estrin, D.: Fine-Grained Network Time Synchronization using Reference Broadcasts. Proceedings OSDI, 2002.

[GWL1] Gao, Y., Wu, K., Li, F.: Analysis on the Redundancy of Wireless Sensor Networks. Proceedings WSNA, 2003.

[HL1] Hsin, C., Liu, M.: Network Coverage Using Low Duty-Cycled Sensors: Random and Coordinated Sleep Algorithms. Proceedings IPSN, 2004.

[HT1] Huang, C., Tseng, Y.: The Coverage Problem in a Wireless Sensor Network. Proceedings WSNA, 2003.

[MKQP1] Meguerdichian, S., Koushanfar, F., Qu, G., Potkonjak, M.: Exposure In Wireless Ad-Hoc Sensor Networks. Proceedings MOBICOM, 2001.

[PCB1] Priyantha, N.B., Chakraborty, A., Balakrishnan, H.: The Cricket Location-Support System. Proceedings MOBICOM, 2000.

[ROB1] Robosoft Advanced Robotics Solutions: http://www.robosoft.fr/SHEET/02Local/1001LAUN/LAUN.html. Last visited on 09/07/2005.

[TG1] Tian, D., Georganas, N.D.: A Coverage-Preserving Node Scheduling Scheme for Large Wireless Sensor Networks. Proceedings WSNA, 2002.

[XBOW1] XBOW Inc.: http://www.xbow.com/Support/Support_pdf_files/MPR-MIB_Series_Users_Manual.pdf. MPR/MIB User's Manual, Last visited on 06/30/2005.

[YHS1] Yan, T., He, T., Stankovic, J.A.: Differentiated Surveillance for Sensor Networks. Proceedings SenSys, 2003.

[YZLZ1] Ye, F., Zhong, G., Lu, S., Zhang, L.: Energy Efficient Robust Sensing Coverage in Large Sensor Networks. UCLA Technical Report, 2002.

A Cross-Layer Approach to Heterogeneous Interoperability in Wireless Mesh Networks

Shih-Hao Shen[1], Jen-Wen Ding[2], and Yueh-Min Huang[1]

[1] Department of Engineering Science National Cheng Kung University,
Tainan 701, Taiwan, ROC
{n9892110, huang}@{ccmail, mail}.ncku.edu.tw
[2] Department of Information Management,
National Kaohsiung University of Applied Sciences,
Kaohsiung 807, Taiwan, ROC
jwding@cc.kuas.edu.tw

Abstract. Routing in a wireless mesh network is a heterogeneous interoperability problem. First, it is possible that some users may disable the relaying functions of their mobile devices in order to save the computing power and battery charge. This will results in heterogeneous relaying capabilities. Second, due to the absence of standard layer-3 ad-hoc routing protocols, various devices may employ different layer-3 routing protocols, making routing difficult. A trivial solution to the above two problem is to flood packets through the un-routable regions to reach the destination region. However, flooding is a brute force approach and will make a broadcast storm, resulting in low throughput of the whole network. In this paper, we propose a cross-layer approach to solve the above problem. Our analysis results show that the proposed cross-layer approach can efficiently provide interoperability without causing broadcast storm.

1 Introduction

There has been a big interest in commercial wireless mesh networks in the recent past. Wireless mesh networks [1] and [2] are ad hoc wireless networks formed to provide an alternate communication infrastructure for mobile and fixed users, in addition to cellular networks. Compared to traditional infrastructure-based networks, wireless mesh networks can be easily deployed and the topology can be easily and rapidly changed via self-organization. Hence wireless mesh networks provide the most economical data transfer capability with support for user mobility [8] and [9]. Many wireless mesh networks have been installed around the world. For example, Nortel networks wireless mesh network solution has been adopted in Taipei city to offer highly scalable, cost-effective last-mile communication to end users [12]. Figure 1 shows the architecture of a wireless mesh network.

Heterogeneous interoperability in a commercial wireless mesh network is an important design issue necessary to be addressed for two reasons. First, since relaying

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 651 – 662, 2005.
© IFIP International Federation for Information Processing 2005

takes up nodes' resources such as computing power and battery charge, some users may disable the forwarding functions of their mobile devices, and by doing so we cannot ensure a fully connected network even with high node density. Research has developed numerous layer-3 routing protocols for ad hoc wireless networks [5], [7], [10] and [11]. Though these protocols find efficient routing path for mobile nodes with minimum control overhead, most of them assume homogeneity of mobile nodes, i.e., all nodes support the full functions of a certain layer 3 routing protocol. This, however, is not the case in a commercial wireless mesh network, where some nodes may disable the relaying functions. In practice, the mobile nodes are classified into two types, the ones enabling the relaying functions and the ones disabling the relaying functions. The second reason is that due to the absence of standard layer-3 routing protocols, various devices may employ different layer-3 routing protocols.

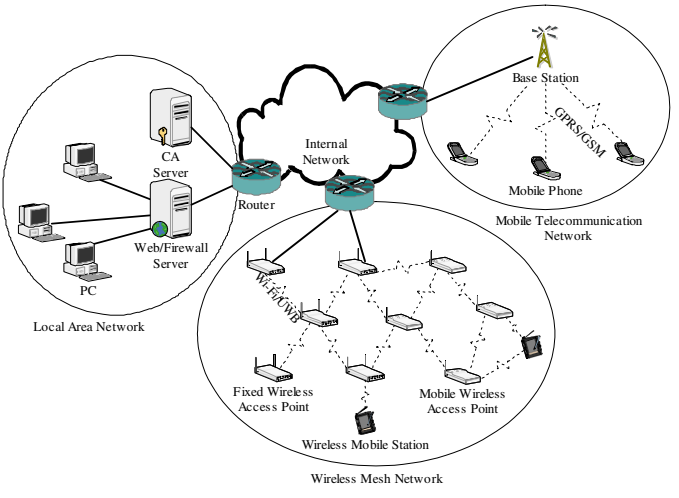


Fig. 1. Architecture of wireless mesh network [8]

Because of the two reasons mentioned above, the whole wireless mesh network can thus be partitioned into many small geographically separated regions belonging to one of the two types. In this paper, the nodes disabling the forwarding function are referred to as un-routable nodes. The nodes using different layer-3 routing protocols are also un-routable nodes for each other. The regions consisting of un-routable nodes are referred to as un-routable regions. It is noted that a region cannot communicate with its un-routable neighbor regions via layer-3 routing.

There are three type of approaches to solving the problem of un-routable regions, (1) the dual stack approach, (2) the naive layer-2 broadcast approach and (3) the cross-layer approach. The dual stack approach employs devices that support more than two protocols and act as a bridge to transform protocols. However, this approach is costly and impractical because of the limited computing power of most mobile devices.

The naive layer-2 broadcast approach floods packets through the un-routable regions to reach the destination region. However, flooding is a brute force approach because it implicates the inability to acquire the whole regional topology of un-routable area. For example, node *d* is a source and node *k* is the destination as shown in Figure 2a. The optimal solution is path *d-c-e-h-k*. Both of the two are routable nodes but separated apart from un-routable regions. Thus, source node (*d*) has no choice but to flood frames. The flooding approach results too many meaningless messages at the MAC layer. As figure 2b illustrated flooding is a very inefficient scheme.

In this paper, we propose a cross-layer approach which consists of a slightly modified version of a common layer-2 protocol, DFWMAC, and a light-weight layer-3 protocol that can cooperate with various layer-3 routing protocols so as to allow the packets to be forwarded without causing a layer-2 broadcast storm. We organize the paper as follows: Section 2 describing related work and providing preliminaries, Section 3 detailing our proposed cross-layer protocol, Section 4 presenting the simulation results, and finally, Section 5 concluding.

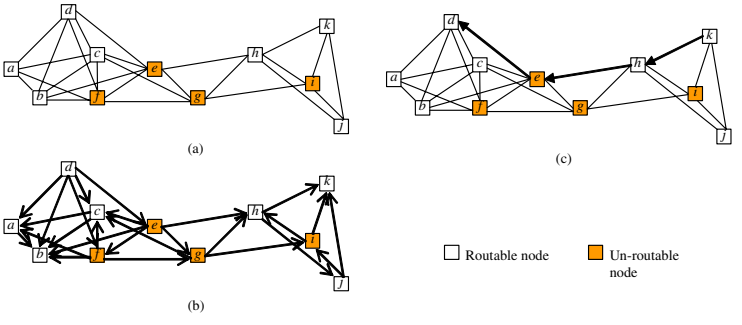


Fig. 2. Using flooding scheme to go through un-routable regions from node *d* to *k*. (a) Connection diagram. (b) Node *d* broadcasting a *search* command. (c) Node *k* responding to a *connect* command.

2 Preliminaries

In this paper, we assume that users have a common MAC layer protocol. In the recent past, numerous MAC protocols have been proposed for ad hoc wireless networks, such as MARCH[13], D-PRMA[4] and DPS[6]. Some of them can even provide guaranteed QoS delivery, such as D-PRMA, DPS. For the sake of compatibility, the proposed cross-layer approach adopts and slightly modifies DFWMAC (Distributed Foundation Wireless Medium Access Control) [3] as its MAC protocol, the DLC (distributed link conjunction) protocol. DFWMAC is the MAC protocol used in IEEE 802.11 recommended standard for wireless ad-hoc and infrastructure LANs. In this section, we briefly review the main operations of DFWMAC that are relevant to our proposed scheme.

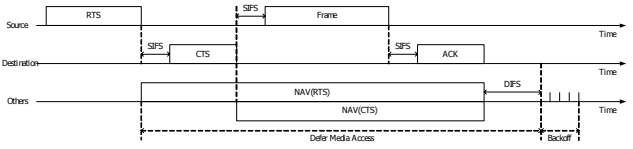


Fig. 3. RTS/CTS access mechanism in DCF

In DFWMAC, Carrier Sense Multiple Access with Collision-Avoidance (CSMA/CA) is used to combat the hidden terminal problem. The MAC coordination function is based on the distributed coordination function (DCF) in ad hoc mode, which utilizes CSMA/CA. CSMA/CA takes advantage of request-to-send (RTS) and clear-to-send (CTS). DCF is a four-way sender-initiated protocol and the most popular collision-avoidance scheme between a pair of sender and receiver. Before source node will send that check medium idle or busy. If medium was busy then use orderly exponential backoff algorithm to avoid collision. Four-way sender-initiated protocol is termed RTS-CTS-DATA-ACK or DFWMAC. As Figure 3 shown, a source node sends a RTS to a destination node. Receiving the RTS, the destination node sends back a CTS. After receiving the CTS, the source node then sends a DATA to destination node, which sends back a ACK. After the process, entry backoff window, any ready sending node contends time slots of backoff.

3 Cross-Layer Protocol

The proposed cross-layer protocol consists of two schemes, the DLC (distributed link conjunction) scheme and the GEER (group entry and exit register) scheme. DLC slightly modifies DFWMAC, the MAC protocol used in IEEE 802.11 recommended standard for wireless ad-hoc and infrastructure LANs.

3.1 Distributed Link Conjunction (DLC) Scheme

As mentioned earlier, in the naive layer-2 approach, link connectivity from a routable region to a destination region via un-routable regions can be achieved by performing flooding in DFWMAC. As shown in Figure 2c, node *d* is a source and node *k* is the destination. Both node *d* and *k* are routable nodes but separated apart from un-routable regions. Source node just simply floods frames. The optimal solution is path *d-c-e-h-k*.

For the sake of compatibility, we design DLC by adopting and slightly modifying the well analyzed and verified MAC protocol used in ad hoc networks, DFWMAC. As shown in Figure 4, DLC uses the same basic RTS-CTS-DATA-ACK frame exchange mechanism. DLC uses piggybacked information onto RTS-CTS handshake and DATA-ACK. The piggybacked information includes (1) source routing address, (2) destination routing address, (3) a down/up stream indication bit.

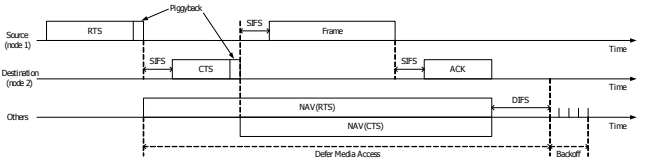


Fig. 4. Piggybacking on DFWMAC

Before data transmission, the sender transmits a *link discovery frame* using the RTS-CTS-DATA-ACK frame exchange mechanism. This link discovery frame will be flooded into the un-routable regions using broadcast (by setting the destination field as a broadcast address). All mobile nodes in the un-routable regions receiving the piggybacked information will perform three operations: (1) rebroadcast the link discovery frame, (2) temporarily cache the information, (3) starts a timer. By checking the cache, redundant link discovery frames received by a node will be discarded. The cache maintains four fields for a link discovery frame: (1) source routing address, (2) destination routing address, (3) a down/up stream indication bit, and (4) the MAC address of the previous sending node of the link discovery frame. Because of the multiple paths, the destination node may receive multiple link discovery frames, but it only acknowledges a *link confirmation frame* for the first received link discovery frame (the first received frame usually implies a shorter path). The link confirmation frame carries information similar to that carried in the link discovery frame. The link confirmation frame is then sent back to the sender node, via the reverse route that thefirst received link discovery frame uses, in a hop-to-hop unicast fashion to avoid broadcast storm. The intermediate un-routable nodes on the reverse path will also cache the information carried in link confirmation

Table 1. The MAC layer protocol

***DLC* Algorithm: (Working on layer-2)**

Signature:

Input: $receive(frame_j)_{j,i}$, $j \in nbrs$

Output: $Send(frame_i)_{i,k}$, $k \in nbrs$

State:

A piggyback $p := (IP_{source}, IP_{distination}, state=\{req|rep\})$

For every $1 \leqq i \leqq n$, $n \in$ {nodes disabled layer-3 functions}

$Node_i$ create a set C_i

$C_i := \phi$

An element of C_i is $e := (p, MAC_{downlink}, MAC_{uplink})$

Tasks:

Periodically maintains C_i by Least-Recently-Used (LRU) replacement strategy.

Overhear neighborhood rebroadcast lead to link state finish.

Transitions:

$receive(frame_j)_{j,i}$

Effect:

if $\exists frame_j.p \in C_i^{\{p\}}$ then

discard $frame_j$

else

if $\exists frame_j.p_{state=rep} \in C_i^{\{pstate=req\}}$ then

send $(frame_j)_{i,\ downlink}$

$MAC_{uplink} := j$

else

send $(frame_j)_{i,\ null} \equiv broadcast(frame_j)_{i,\ \forall k}$

$MAC_{downlink} := j$

$send(frame_i)_{i,k}$

Effect:

if $k ==$ *null* then

$broadcast(frame_i)$

else

$unicast(frame_i)_k$

frame. A node receiving link discovery frame but not receiving the corresponding link confirmation frame will delete the cache information after the previously set timer is matured. The MAC layer protocol is presented in table 1.

Figure 5 shows a simple example of node *a* communicating to node *b*. Nodes *a*, *b* and *c* are routable nodes. Node *1* and *2* are un-routable nodes. Nodes *a* and *b* communicate via layer-3 routing protocol. Nodes *b* and *c* communicate by DLC. Figure 5a shows how link discovery frame is broadcasted and how intermediate un-routable nodes cache the information carried in the link discovery frame. Figure 5b shows how link confirmation frame is unicasted to the sender node via the reverse path and how intermediate un-routable nodes cache the information carried in the link discovery frame.

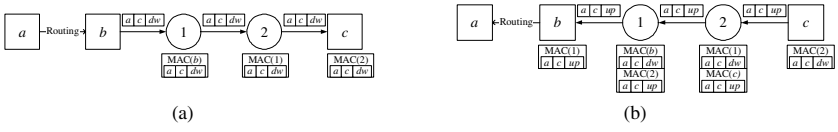


Fig. 5. Example of a caching frame to reserve paths

3.2 GEER Routing Protocol

In the recent past, numerous routing protocols have been proposed for ad hoc wireless networks, such as DSDV[10], WRP[7], DSR[5] and AODV[11]. As mentioned earlier, these protocols cannot solve the interoperability problem. We consider a scenario where heterogeneous nodes are un-routable for each other because some of them may disable the forwarding functions or because they may employ different routing protocols, as shown in Figure 2a. The network is therefore partitioned into many small geographically separated regions. In a routable region, a packet can be directly sent any other node in the same region using a common layer-3 routing protocol, such as DSR or AODV. However, when a packet is sent to a destination that is in an un-routable region, a special design will be required.

Our proposed cross-layer approach employs DLC with very limited broadcast to pass packets through an un-routable region. However, two problems may arise if only DLC is used. First, the layer-2 link discovery frame of DLC may result in a routing cycle. Second, when a packet must travel through several routable and un-routable regions, in order to efficiently relay packets in a routable region without flooding, each region must perform smart relaying. The two problems are illustrated in Figure 6. To cope with the two problems, the proposed cross-layer approach uses a light-weight layer-3 routing protocol, Group Entry and Exit Register (GEER), that can cooperate with existing routing protocols for ad hoc wireless networks, such as DSR and AODV.

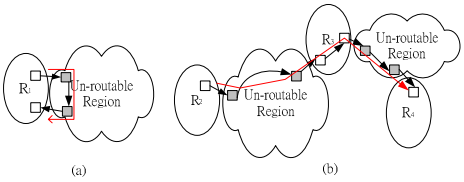


Fig. 6. Effect of routing on wireless mesh network interoperability: (a) Routing cycle. (b) Multi-regions forwarding.

In GEER protocol, three types of nodes are introduced. To prevent routing cycle and to help crossing multi-regions routing we need a node to record the entry and exit of link discovery frame of DLC. The special node is termed *GEER node*. GEER is elected from a group of nodes. To concatenate different routable regions via DLC, the nodes, termed *dam nodes*, sit between a routable and an un-routable region must perform the relaying function. A special un-routable node whose routable neighbors are in the same routable region is referred to as *surrounded node*.

The algorithm is to determine whether the node is GEER by searching a maximum degree node in a region. It is easier when routing protocol is pre-active. Dam nodes and

Table 2. DAM algorithm

***DAM* Algorithm: (Working between layer-2 and layer-3)**	
Signature:	**Transitions:**
Input: Output:	$send(packet_t)_{i,k}$
$receive(packet_j)_{j,i}$, $j \in$ *Homo-Network* $Send(packet_t)_{i,k}$, $k \in$ *Homo-Network*	Effect:
State:	$send(packet_t)_{i,k}$
Broadcast storm avoidance set $\boldsymbol{A}:=\varnothing$	
Tasks:	$receive(packet_s)_{j,i}$, s is source node
Precondition:	Effect:
For every n $\in H_i$, *determine adapting to a dam node*	*if* $packet_s ==$ *GEERtoDam then*
Nodes of homo-group H_i *elect a* $GEER_i$.	*DLC.send()*
Effect:	*if* $packet_s \in$ *cmd then*
Periodically deletes timeout element of $\boldsymbol{A}$.	*For every* $d \in D_i$
$\forall$*Destination* $j \in H_i$, $send(packet_t)_{i,j}$	$send(GEERtoDam(packet_s \cup \{IP_{GEERi}\}))_{i,d}$
$\forall$*Destination* $k \notin H_i$, $send(Send\ (packet_t)_{i,k})_{i,GEERi}$	$\boldsymbol{A}:=\boldsymbol{A} \cup \{ packet_s + t\}$, *t is a timestamp*
	else
	Run original routing protocol
	else $s \notin H_i$
	if $packet_s \in$ *cmd then*
	if i is distination then
	$send(GEERtoDam(rep_packet_j))_{i,j}$
	else i is not distination
	For every $d \in D_i$
	$send(GEERtoDam(packet_s \cup \{IP_{GEERi}\}))_{i,d}$
	$\boldsymbol{A}:=\boldsymbol{A} \cup \{ packet_s + t\}$, *t is a timestamp*
	else $packet_s \notin$ *cmd*
	$send(GEERtoDam(rep_packet_j))_{i,j}$

surrounded nodes are determined as follows table 2. Every node (routable or un-routable) stores the MAC address of adjacent un-routable nodes. The GEER of a routable region keeps a GURID (Group Un-routable Identifier) table, which records the MAC and IP addresses of each routable node in the region and the MAC addresses of

these nodes' adjacent un-routable nodes. If a routable node wants to detect if an adjacent un-routable node is a surrounded un-routable node, it sends a query with the MAC address of the un-routable node to GEER. If the query of GURID table shows that the un-routable node is adjacent to all routable nodes, then it is a surrounded node. A routable node can query the GEER to determine if it is a dam node by sending the MAC addresses of its adjacent nodes. If not all adjacent nodes are surrounded nodes, this routable node is a dam node. As shown in Figure 7, nodes *h*, *k*, and *j* are routable nodes and nodes *e*, *g* and *i* are un-routable nodes. Node *i* is a surrounded un-routable node because all node *i*'s adjacent MAC addresses appear in the GURID. Nodes *h* is a dam node because it is adjacent to out-region nodes *e* and *g*.

Table 3. GEER algorithm

***GEER* Algorithm: (Working on layer-3)**	
Signature:	**Transitions:**
Input: Output:	$send(packet_i)_{i,k}$
$receive(packet_j)_{j,i}$, $j \in$ *Homo-Network* $Send(packet_i)_{i,k}$, $k \in$ *Homo-Network*	Effect:
State:	$send(packet_i)_{i,k}$
Broadcast storm avoidance set $\boldsymbol{A} := \varnothing$	
Command set cmd={"Send", "GEERtoDam"}	$receive(packet_s)_{j,i}$, s is source node
Tasks:	Effect:
Precondition:	*if* $s \in H_i$ *then*
For every n $\in H_i$, *determine adapting to a dam node*	*if* $packet_s \in$ *cmd then*
Nodes of homo-group H_i *elect a* $GEER_i$.	*For every* $d \in D_i$
Effect:	$send(GEERtoDam(packet_s \cup \{IP_{GEERi}\}))_{i,d}$
Periodically deletes timeout element of $\boldsymbol{A}$.	$\boldsymbol{A} := \boldsymbol{A} \cup \{ packet_s + t \}$, *t is a timestamp*
$\forall$*Destination* $j \in H_i$, $send(packet_i)_{i,j}$	else
$\forall$*Destination* $k \notin H_i$, $send(Send\ (packet_i)_{i,k})_{i,GEERi}$	*Run original routing protocol*
	else $s \notin H_i$
	if $packet_s \in$ *cmd then*
	if i is distination then
	$send(GEERtoDam(rep_packet_i))_{i,j}$
	else i is not distination
	For every $d \in D_i$
	$send(GEERtoDam(packet_s \cup \{IP_{GEERi}\}))_{i,d}$
	$\boldsymbol{A} := \boldsymbol{A} \cup \{(packet_s, t)\}$, *t is a timestamp*

We then discuss the concept of GEER routing algorithm shown in Table 3. GEER is a centralized control in each routable region. Every packet sent out and sent in the region needs to register to GEER. By storing and comparing packets to see if it is repeatedly received, GEER can avoid a routing cycle mentioned earlier. If a destination node does not exist in the region, GEER forwards packets to its dam nodes to go through un-routable regions to the destination node. As shown in Figure 7, consider a simplified example that node *d* needs to communicate to node *k*. Initially, node *d* sends a *GEER routing query* to *c*. Because node *k* does not exist in *c*'s region, such as *c* multicasts the query to its dam nodes *c* and *d* to go through un-routable region. In the first path, a routing cycle *c*-*e*-*d* occurs. When *d* receives a frame with piggyback, it obtains the destination identifier and queries it by GEER *c*. The GEER finds it is a redundant query packet and then returns the command of "discard" to *d*. In the second path, *c*-*e*-*h*-*k*, when receiving a frame with piggyback, *k* obtains the destination identifier and queries it by GEER *e*.

It is noted that the proposed cross-layer scheme can cooperate with other nodes not supporting GEER and therefore maintains interoperability in wireless mesh network as Figure 7 shown. There is at least one node that supports GEER, however. Otherwise, there might be some independent regions that impede routing.

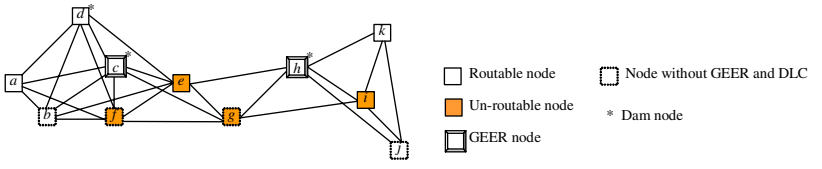


Fig. 7. A GEER operating snapshot in figure 2, where partial network lacks GEER and DLC operation

4 Performance Evaluation

In order to verify the cross-layer protocol, we conduct a series of simulations using the Distribution Wireless Mesh Network Simulator developed by ourselves. We first investigate how heterogeneous nodes will be distributed over an area. We perform a set of simulations for heterogeneous node distribution, which ranges from 50 to 450 nodes (with increase of 50) spreading randomly in a network area of 1500×1500 meter^2. We

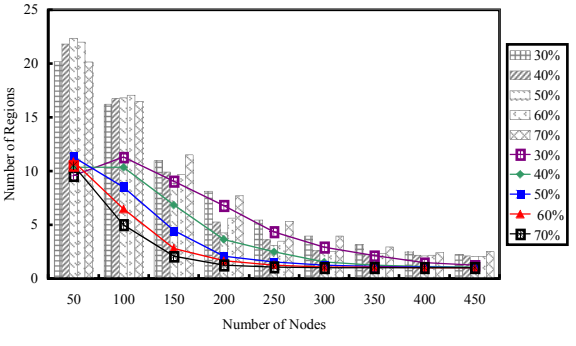


Fig. 8. Heterogeneous case – when the density of routable nodes are 30%, 40%, 50%, 60% and 70%: Average number of routable regions (frequency polygon) and average number of total regions (histogram)

use random distribution on two-dimension space. To take into account the fact that mobility may affect linking topology, we assume that renew process happens prior to rush hour everyday. We simulate different density of routable nodes' from the snapshots as renew and ignore mobility.

We study the effect of the ratios of nodes with and without cross-layer protocol on connection regions. In this experiment, the simulated topology consists of only two types of nodes, routable and un-routable nodes. As we can observe from the simulation result shown in Figure 8, what mainly affects the amount of routable regions is the number of routable nodes. The number of routable nodes is over 150 in the set environment, and this almost leads to a complete routable region. Hence we set the parameters for the cross-layer protocol as follows: the densities of routable nodes are 40%, 50%, and 60%, and the total numbers of nodes are 100, 150, 200, and 250, respectively. We evaluate overhead of routing control, and the result is given in Figure 9a. The result shows that, when we use flooding to reach destination nodes, the amount of control packets dramatically increases with the increasing numbers of nodes. In our cross-layer protocol, most of the control packets come from the first time discovery. Given a certain amount of routable nodes in the environment, however, it takes much fewer control traffic overhead to complete routing when our cross-layer protocol is used. Next, we investigate the routing delay time and the result is given in Figure 9b. Our cross-layer approach does not perform better than the simple layer-2 flooding approach because it takes more time to perform the layer-3 algorithm. However, we can observe figure 9 that the cross-layer protocol is more beneficial because the number from source to destination takes only 4 hops in our simulation environment.

Table 4. Simulation parameters

Network area size	1500x1500 m^2
Transmission radius	225 m
Transmission Rate	11 Mbps
Avg. frame size	64 bytes
Avg. packet size	512 bytes
Avg. Routing discovery	3 (normal distribution, standard deviation: 3)
Speed	0~1.41 m/s (random walk)
Simulation Time	300 Sec

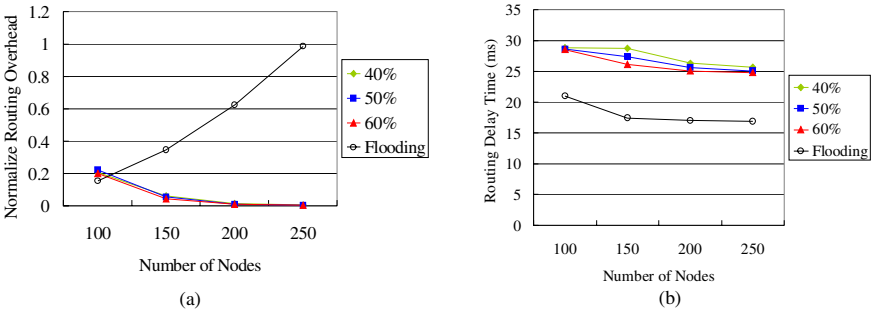


Fig. 9. Heterogeneous case - 40%, 50% and 60% routable nodes: (a) Normalized routing overhead vs. number of total nodes. (b) Route discovery delay vs. number of total nodes.

5 Conclusion

In a commercial wireless mesh network, interoperability among heterogeneous mobile devices is a difficult design issue. Firstly, it is possible that some users may disable the relaying functions of their mobile devices to save the computing power and battery charge. This phenomenon results in heterogeneous relaying capabilities. Secondly, due to the absence of standard layer-3 routing protocols, various devices may employ different layer-3 routing protocols. A naive approach to the interoperability problem is to flood traffic through un-routable regions via a common MAC protocol. However, this will result in a broadcast storm, yielding a low throughput of the whole network. In this paper, we propose a cross-layer approach to addressing this problem. The cross-layer protocol consists of two key components: (a) DLC, a MAC protocol that effectively restrains the broadcast storm occurring in the un-routable regions, (b) GEER, a light-weight routing protocol that uses Group Entry and Exit Register node to avoid routing cycles and to help multi-region relaying. It is worth mentioning that DLC is a slightly modified version of DFWMAC, the MAC protocol used in IEEE 802.11 recommended standard for wireless ad-hoc networks and infrastructure LANs. Our analysis results show that the proposed cross-layer approach can achieve the interoperability goal without causing the broadcast storm problem.

Reference

1. Akyildiz, I.F., Wang, X. and Wang, W.: Wireless Mesh Networks: A Survey. Computer Networks Journal, Vol. 47, 3 (2005) 445-487
2. Ashish, R., Kartik, G. and Chiueh, T. C.: Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks. ACM SIGMOBILE Mobile Computing and Communications Review Vol. 8, Issue 2, 4 (2004)
3. IEEE Computer Society: 802.11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. 7 (1997)
4. Jiang, S., Rao, J., He, D., Ling, X. and Ko, C.C.: A simple distributed PRMA for MANETs. IEEE Trans. Veh. Tech., Vol.51, No.2, 3 (2002) 293-305
5. Johnson, D. B. and Maltz, D. A.: Dynamic Source Routing in Ad-Hoc Wireless Networks. Mobile Computing, 1994.
6. Kanodia, V., Li, C., Sabharwal, A., Sadeghi, B. and Knightly, E.: Distributed multi-hop scheduling and medium access with delay and throughput constraints. ACM/Baltzer Journal of Wireless Networks, Vol. 8, No. 5, 9 (2002) 455-466
7. Murthy, S. and Garcia-Luna-Aceves, J. J. :An Efficient Routing Protocol for Wireless Networks. MONET, Vol.1, No. 2, 10 (1996) 183-197
8. Nortel Networks Corp.: Wireless Mesh Network Solution. http://www.nortelnetworks.com/ solutions/wrlsmesh/ #, (2005)
9. Oreilly Corp.: Wireless Mesh Networking. http://www.oreillynet.com/pub/a/wireless/ 2004/01/22/ wirelessmesh.html, 1 (2001)
10. Perkins, C. E. and Bhagwat, P.: Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. Comp. Commun. Rev., 10 (1994) 234–244

11. Perkins, C. E. and Royer, E. M.: Ad-Hoc On-Demand Distance Vector Routing. In Proc. 2nd IEEE Wksp. Mobile Comp. Sys. and Apps., 2 (1999) 90–100
12. Toh, C. K., Vassiliou, V., Guichal, G. and Shih, C. H.: MARCH: A medium Access Control Protocol for Multi-Hop Wireless Ad Hoc Networks. Proc. of IEEE MILCOM'00, Vol. 1, 10 (2000) 512-516

Reliable Time Synchronization Protocol for Wireless Sensor Networks

Soyoung Hwang and Yunju Baek

Department of Computer Science and Engineering,
Pusan National University, Busan 609-735, South Korea
{youngox, yunju}@pnu.edu

Abstract. Sensor network applications need synchronized time to the highest degree such as object tracking, consistent state updates, duplicate detection, and temporal order delivery. In addition to these domain-specific requirements, sensor network applications often rely on synchronization as typical distributed system do: for secure cryptographic schemes, coordination of future action, ordering logged events during system debugging, and so forth. This paper proposes a Reliable Time Synchronization Protocol (RTSP) for wireless sensor networks. In the proposed method, synchronization error is decreased by creating hierarchical tree with lower depth and reliability is improved by maintaining and updating information of candidate parent nodes. The RTSP reduces recovery time and communication overheads comparing to TPSN (Timing-sync Protocol for Sensor Networks) when there are topology changes owing to moving of nodes, running out of energy and physical crashes. Simulation results show that RTSP has about 10% better performance than TPSN in synchronization accuracy and the number of message in the RTSP is 10%∼35% lower than that in the TPSN when nodes are failed in the network. In case of different transmission range of nodes, the communication overhead in the RTSP is reduced up to 50% than that in the TPSN at the maximum.

1 Introduction

Recent advances in sensors, MEMS (Micro-Electro-Mechanical Systems), low power and highly integrated digital electronics and low power RF technology have allowed the construction of low-cost small sensors nodes. Such sensor nodes are generally equipped with computation and wireless communication capabilities that can form distributed wireless sensor network systems. The sensing circuitry of sensor nodes measures ambient conditions related to the environment and transforms them into an electric signal. Processing such a signal reveals some properties about objects located and/or events happening in the vicinity of the sensor. The sensor node sends such collected data, usually via radio transmitter, to a command center (sink) either directly or through a data concentration center (a gateway). A natural architecture for such collaborative distributed sensor nodes is a network with wireless links that can be formed among the sensor

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 663–672, 2005.
© IFIP International Federation for Information Processing 2005

nodes in an ad hoc manner [1,2]. The most important characteristic of these sensor networks is the crucial need for energy efficiency. To facilitate easy deployment without an infrastructure, many nodes will necessarily be untethered, having only finite energy reserves from a battery [3]. These sensor networks can be used for various application areas such as health, military, home network, managing inventory, monitoring disaster areas and so on.

The main technology in wireless sensor networks includes hardware platforms and OS, low-energy consumption network protocols, time synchronization, localization, middleware, security, and applications. Especially, distributed wireless sensor networks make particularly extensive use of synchronized time: for example, to integrate a time-series of proximity detection into a velocity estimate; to measure the time-of-flight of sound for localizing its source; to distribute a beamforming array; or to suppress redundant messages by recognizing that they describe duplicate detections of the same event by different sensors. In addition to these domain-specific requirements, sensor network applications often rely on synchronization as typical distributed system do: for secure cryptographic schemes, coordination of future action, ordering logged events during system debugging, and so forth [4,5]. Since the characteristic of sensor nodes with limited computation and energy, traditional time synchronization protocols in distributed systems can not be applied to the sensor networks directly. So existing synchronization methods are revised or new approaches are proposed to synchronize the sensor networks.

In this paper we propose reliable time synchronization protocol for wireless sensor networks. It constructs hierarchical topology in the first phase, and performs pair-wise synchronization in the second phase. In the proposed method, synchronization error is decreased by creating hierarchical tree with lower depth and reliability is improved by maintaining and updating information of candidate parent nodes. The RTSP reduces recovery time and costs - communication overhead - comparing to TPSN [6] when there are topology changes owing to moving of nodes, running out of energy and physical crashes.

The rest of this paper is organized as follows. Section 2 discusses motivation and related research in the area. In section 3, we describe proposed time synchronization protocol for wireless sensor networks. Section 4 includes the performance evaluation of the proposed method. Finally, we conclude this paper in section 5.

2 Motivation and Related Work

Since the characteristic of sensor nodes with limited computation and energy, traditional time synchronization protocols in distributed systems can not be applied to the sensor networks directly. So existing synchronization methods are revised or new approaches are proposed to synchronize the sensor networks. In the first stage of research on time synchronization in sensor networks, most approaches are based on the synchronization model such as event ordering or relative clock. These methods do not synchronize the sensor node clocks but

generate a right chronology of events or maintains relative clock of nodes. From a viewpoint of network topology, synchronization coverage is limited in a single broadcast domain; however, typical wireless sensor networks operate in areas larger than the broadcast range of a single node, so network-wide time synchronization is needed essentially. Besides, adjusting the local clock has better efficiency than maintaining relative clock since it requires more memory capacity and communication overheads. TPSN and FTSP are the representative ones which meet these requirements [7].

TPSN works in two phases: level discovery and synchronization. The aim of the first phase is to create a hierarchical topology in the network, where each node is assigned a level. Only one node is assigned level 0, the root node. In the second phase, a node of level i synchronizes to a node of level i-1. At the end of the synchronization phase, all nodes are synchronized to the root node, and network-wide synchronization achieved [6].

The goal of the FTSP is to achieve a network wide synchronization of the local clocks of the participating nodes. The assumptions in FTSP are that each node has a local clock exhibiting the typical timing errors of crystals and can communicate over an unreliable but error corrected wireless link to its neighbors. The FTSP synchronizes the time of a sender to possibly multiple receivers utilizing a single radio message time-stamped at both the sender and the receiver sides. It compensates for the relevant error sources by utilizing the concepts of MAC layer time-stamping and skew compensation with linear regression [8].

FTSP achieves robustness against node and link failures by utilizing periodic flooding of synchronization message and implicit dynamic topology update. On the other hand, TPSN does not handle dynamic topology changes; however, FTSP can not be applied generally since the synchronization accuracy in FTSP is seriously affected by the analyzed source of delays and uncertainties which are varied according to changes of the systems. The synchronization accuracy of network-wide multi-hop synchronization is a function of the construction and depth of the tree. The synchronization error is propagated hop by hop. Therefore new approaches are required to reduce the synchronization error and to manage dynamic topology changes.

3 Reliable Time Synchronization Protocol

In the following we present our scheme called *Reliable Time Synchronization Protocol* (RTSP) for wireless sensor networks. As mentioned before, the synchronization accuracy is a function of the construction and depth of the tree in network-wide multi-hop time synchronization. So it is necessary that every node is assigned a level with the shortest path from the root node to reduce synchronization error. Besides, sensor nodes can fail easily such as nodes may move, may run out of energy and may be destroyed physically. Hence a scheme is needed to manage nodes failure for synchronization accuracy and energy efficiency. We designed the protocol considering these issues.

3.1 Basic Concept

The proposed reliable time synchronization protocol works in two phases. It is assumed that nodes in the network have unique ID. But it does not need that each node is aware of the neighbor set as in the TPSN. The management of neighbor nodes is included in the operations of the protocol.

In *the first phase – hierarchical topology setup* – a hierarchical topology is created in the network. Root node with level *0* initiates topology setup. A node receives topology setup messages and assigns its level by selecting a parent with lowest level to reduce the depth of tree in the network. Other parent information is stored in candidate parent list for node failure management. Eventually every node is assigned level and a tree structure is constructed.

In *the second phase – synchronization and handling topology changes* – a node belonging to level *i* synchronizes to its parent node which is belonging to level *i-1* by exchanging time-stamp messages. When a node can not communicate with its parent, it selects another parent in the candidate list and performs synchronization. If the candidate list is empty, it request level setup to its neighbors and assigns new level, new parent and candidate parents. Candidate list is updated periodically by listening to communications of neighbors.

Following figure 1 and formula show how to obtain clock offset of a node which is used widely in many time synchronization protocols. Clock offset represents the amount to adjust the local clock to bring it into correspondence with the reference clock.

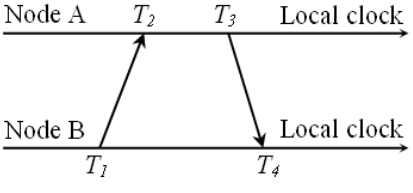


Fig. 1. Measuring delay and offset

As in the NTP, the roundtrip delay and clock offset between two nodes A and B are determined by a procedure in which timestamps are exchanged via wireless communication links between them. The procedure involves the four most recent timestamps numbered as show in figure 1. The T_1 and T_4 timestamps are determined relative to the node A clock, while the T_2 and T_3 timestamps are determined relative to the node B clock. The measured roundtrip delay δ and clock offset θ of B relative to A are given by [9]

$$\delta = (T_4 - T_1) - (T_3 - T_2)\,,\ \theta = \frac{(T_2 - T_1) + (T_3 - T_4)}{2}.$$

3.2 Protocol Description

The proposed reliable time synchronization protocol works in two phases. Operations of the protocol are detailed as follows.

The first phase: Hierarchical topology setup
In the first phase, a hierarchical topology is created in the network. This phase enforces to create a tree structure with lower depth and candidate parent list is generated to manage failure of nodes in the network.

Step 1: The root node initiates topology setup phase. Level *0* is assigned to the root node. It broadcasts topology setup message with its ID and its level.

Step 2: A node receives topology setup message during pre-defined time interval. (Root node discards this message.) It selects a parent with the lowest level number from received messages and stores other information to the candidate parent list according to the level number. Then it broadcasts topology setup message with its ID and its level.

Step 3: Each node in the network performs step 2 and eventually every node is assigned level.

Step 4: When a node does not receive topology setup message or a new node joins the network, it waits for some time to be assigned a level. If it is not assigned a level within that period, it broadcasts topology setup request message and then performs step 2 with reply of its neighbors.

The second phase: Synchronization and handling topology changes
In the second phase, a node belonging to level i synchronizes to its parent node which is belonging to level *i-1* by exchanging time-stamp messages. When a node can not communicate with its parent, it selects another parent in the candidate list and performs synchronization.

Step 1: The root node initiates synchronization phase by broadcasting synchronization message.

Step 2: On receiving synchronization message, nodes belonging to level *1* exchange time-stamp message with the root node and adjust the local clock and then broadcast synchronization message.

Step 3: On receiving synchronization message, each node belonging to level i exchanges time-stamp message with its parent and performs step 2. Eventually every node is synchronized. Once it receives a synchronization message, it discards additional messages from other upper level nodes.

Step 4: When a node can not communicate with its parent, it selects another parent in the candidate list, updates own level - if it is needed - and performs step 3. The level of its child nodes will be updated when they execute synchronization. If the candidate list is empty, it performs step 4 of the topology setup phase ahead. Candidate list can be updated periodically by listening to communications of neighbors.

When the root node fails, a node which has the lowest ID in the next level takes over it. The synchronization accuracy may be improved by utilizing the concepts of MAC layer time-stamping as in the TPSN, and the random back-off mechanism can be adapted to avoid the collision of wireless links.

4 Performance Evaluation

In order to evaluate the performance of the proposed method, we established a simulation model in the NESLsim based on the PARSEC platform. PARSEC (PARallel Simulation Environment for Complex systems) is a C-based discrete-event simulation language. It adopts the process interaction approach to discrete-event simulation [10]. In NESLsim, a sensor network is modeled as a collection of sensor nodes, a channel, and a supervising entity to create the nodes [11].

4.1 Simulation Model

N nodes are deployed in a uniformly random fashion over a sensor terrain of size 100x100. Each node has a transmission range of 28. The number of nodes, N, is varied from 100 to 300 with each increase of 50. All other parameters are arranged with the same value in the TPSN simulation. The setup includes a CSMA MAC. The radio speed is 19.2kb/s, similar to the UC Berkeley MICA Motes, and every packet has a fixed size of 128bits. A node is chosen randomly to act as the root node. The granularity of the node clocks, which is the minimum accuracy attainable, is 10s. The clock model used in simulations has been derived from the characteristics of the oscillators used in sensor nodes. The frequency drift is varied randomly with time, within the specified range, to model the temporal variations in temperature. All sensor node clocks drift independently of each other. There is an initial random offset uniformly distributed over 2 seconds among the sensor node clocks to capture the initial spatial temperature variations and the difference in the boot up times [12].

4.2 Simulation Results

All results are averaged over hundred simulation runs. The performance is compared to TPSN. The depth of a tree means the length of the path from the root node to a node with the highest level number. The synchronization error is defined as the difference between the clocks of the sensor nodes and the root node.

Figure 2 shows average depth of the tree per nodes. The synchronization accuracy of multi-hop synchronization is a function of the construction and depth of the tree. So the lower depth of tree has the better synchronization accuracy. Usually RTSP has 1∼2 lower depth than TPSN.

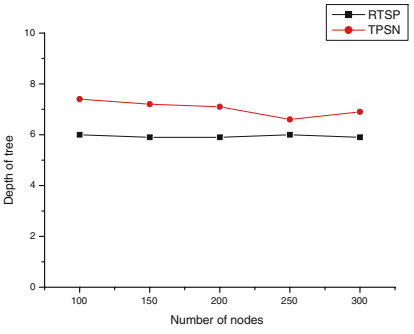
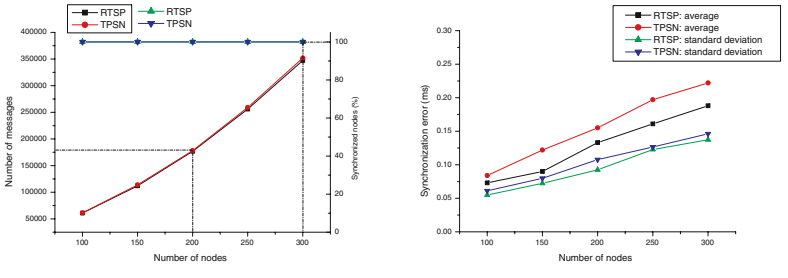


Fig. 2. Depth of tree

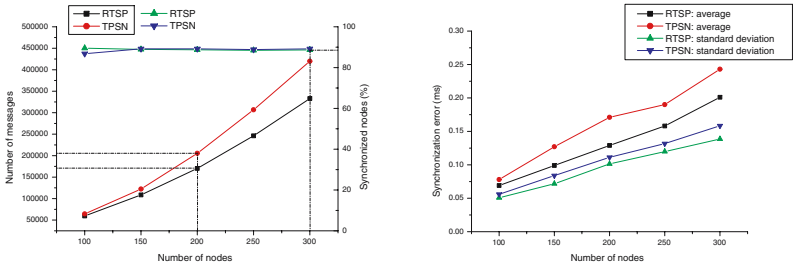
In figure 3, the number of messages processed during the simulation and the synchronization accuracy are presented when there is no failure of nodes. In the almost same number of messages, the RTSP has better performance in synchronization accuracy. This is the effect of the tree depth.



(a) Number of messages (b) Synchronization error

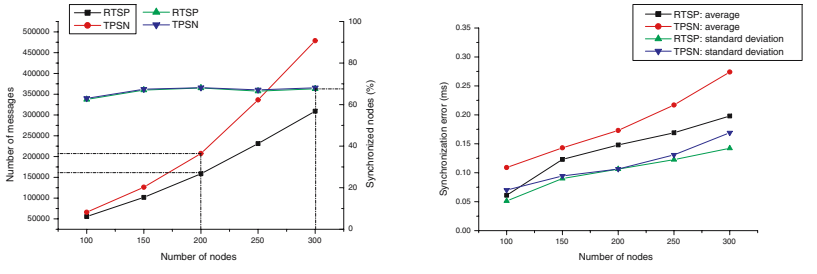
Fig. 3. Without failure of nodes

Figure 4 and figure 5 show the number of messages processed during the simulation, synchronized proportion of nodes and synchronization accuracy when there are 10% and 30% failure of nodes respectively. In sensor networks, sensor nodes can fail easily such as nodes may move, may run out of energy and may be destroyed physically. And this failure of nodes leads to topology changes. In the simulation, node failure means that there are topology changes. In a similar proportion of synchronized nodes to the entire nodes, RTSP reduces the number



(a) Number of messages (b) Synchronization error

Fig. 4. 10% failure of nodes



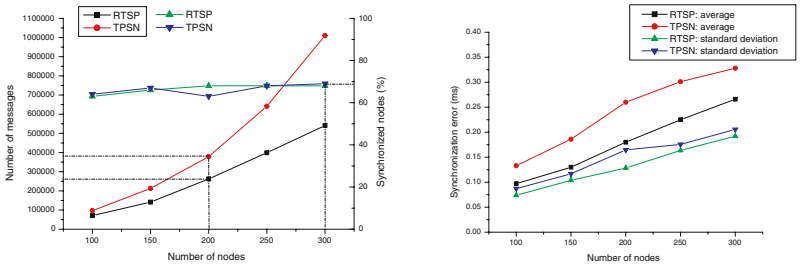
(a) Number of messages (b) Synchronization error

Fig. 5. 30% failure of nodes

of messages and shows better performance in synchronization accuracy. In wireless sensor networks, communication is one of the dominant factors in energy efficiency. Therefore communication overheads must be reduced to save energy. The RTSP reduces the number of messages and improves the synchronization accuracy by handing dynamic topology changes through the candidate parent list.

As can be seen in the results, the performance of RTSP gets better than TPSN as the failure rate (topology changes) is increased. At 10% failure out of 300 nodes, the number of messages in the RTSP is 20% lower than that in the TPSN. At 30% failure out of 300 nodes, the number of messages in the RTSP is decreased by 35% comparing to that in the TPSN.

Additionally, we varied transmission range of nodes. Except transmission range, all other parameters are arranged with the same value as described in the section 4.1 Simulation Model. Each node has different transmission range from 20 to 40. Figure 6 depicts the number of messages processed during the simulation, synchronized proportion of nodes and synchronization accuracy when there is 30% failure of nodes in different transmission range. In case of different transmission range, the number of messages in the RTSP is 25%∼50% lower than that in the TPSN when there are topology changes.



(a) Number of messages (b) Synchronization error

Fig. 6. 30% failure of nodes in different transmission range

5 Conclusions

As in any distributed computer system, time synchronization is a critical issue in sensor networks. Time synchronization is a prerequisite for sensor network applications such as object tracking, consistent state updates, duplicate detection, and temporal order delivery. In addition to these domain-specific requirements, sensor network applications often rely on synchronization as typical distributed system do: for secure cryptographic schemes, coordination of future action, ordering logged events during system debugging, and so forth. But traditional time synchronization methods in distributed systems can not be applied to the sensor networks directly because of the characteristic of sensor networks with limited computation and energy.

In this paper we proposed reliable time synchronization protocol for wireless sensor networks. It constructs hierarchical topology in the first phase, and performs pair-wise synchronization and handling topology changes in the second phase. In the proposed method, synchronization error is decreased by creating hierarchical tree with lower depth and reliability is improved by maintaining and updating information of candidate parent nodes. The RTSP reduces recovery time and costs - communication overheads - comparing to TPSN when there are changes of topology. In order to evaluate the performance of the proposed method, we established a simulation model in the NESLsim based on the PARSEC platform. Simulation results shows that RTSP has about 10% better performance than TPSN in synchronization accuracy. And the number of message in the RTSP is 10%∼35% lower than that in the TPSN when there are topology changes. In case of different transmission range of nodes, the communication overhead in the RTSP is reduced up to 50% than that in the TPSN at the maximum.

Acknowledgment. This work was supported by the Regional Research Centers Program (Research Center for Logistics Information Technology), granted by the Korean Ministry of Education and Human Resources Development.

References

1. Kim, D.:Ubiquitous sensor networks, Real-Time Embedded World 19, pp.34-43, 2004.
2. Akkaya, K. Younis, M.: A survey on routing protocols for wireless sensor networks, Elsvier Ad Hoc Networks Journal 3(3), pp.325-349, 2005.
3. Elson, J., Estrin, D.: Time synchronization for wireless sensor networks, Proceedings of the IEEE International Symposium on Parallel and Distributed Processings, pp.1965-1970, 2001.
4. Elson, J., Romer, K.: Wireless Sensor Networks: A new regime for time synchronization, ACM Computer Communication Review 33(1), pp.149-154, 2003.
5. Elson, J.: Time synchronization in wireless sensor networks, Ph.D. Thesis, UCLA, 2003.
6. Ganeriwal, S. Kumar, R., Srivastava, M.B.: Timing-sync protocol for sensor networks, Proceedings of the ACM SenSys, pp.138-149, 2003.
7. Hwang, S.Y, Baek, Y.J.: A survey on time synchronization for wireless sensor networks, ESLAB Technical Report, 2004.
8. Maroti, M., Kusy, B., Simon, G., Ledeczi, A.: The flooding time synchronization protocol, Proceedings of ACM SenSys, pp.39-49, 2004.
9. Mills, D.L: Network Time Protocol (Version 3) Specification, Implementation and Analysis, RFC1305, 1992.
10. PARSEC User Manual, http://pcl.cs.ucla.edu/projects/parsec, 1999.
11. Ganeriwal, S., Tsiatsis, V., Schurgers, C., Srivastava, M.B.: NESLsim: A parsec based simulation platform for sensor networks, NESL, 2002.
12. Ganeriwal, S., Kumar, R., Adlakha, S., Srivastava, M.B.: Network-wide time synchronization in sensor networks, NESL Technical Report, 2003.

HMNR Scheme Based Dynamic Route Optimization to Support Network Mobility of Mobile Network

Moon-Sang Jeong[1], Jong-Tae Park[1], and Yeong-Hun Cho[2]

[1] School of Electronic and Electrical Engineering, Kyungpook National University 1370, Sankyuk-Dong, Buk-Gu, Daegu, Korea
{msjeong, jtpark}@ee.knu.ac.kr
[2] Department of Information and Communication, Kyungpook National University 1370, Sankyuk-Dong, Buk-Gu, Daegu, Korea
yhcho@ee.knu.ac.kr

Abstract. A lot of recent research has been focused on developing network mobility management to support the movement of a mobile network consisting of several mobile nodes. In the mobile ad-hoc network environment, network itself can be moved to another point. For the network mobility, the IETF NEMO working group proposed the basic support protocol for the network mobility to support the movement of a mobile network consisting of several mobile nodes. However, this protocol has been found to suffer from the so-called 'dog-leg problem', and despite alternative research efforts to solve this problem, there are still limitations in the efficiency for real time data transmission and intra-domain communication. Accordingly, this paper proposes a new route optimization methodology that uses unidirectional tunneling and a tree-based intra-domain routing mechanisms which can significantly reduce delay in both signaling and data transmission.

1 Introduction

As the technology related to the wireless and mobile ad-hoc network environments is rapidly being developed, it increases the necessity for research about network mobility that could support the ad-hoc mobility of not only a single mobile node but also the movement of a mobile ad-hoc network which consists of several mobile nodes [1]. The most representative work is that of the IETF (Internet Engineering Task Force) NEMO (Network Mobility) working group. The IETF NEMO working group has proposed several Internet drafts [2], [3], [4].

The NEMO basic support protocol defines a methodology for supporting network mobility by using bi-directional tunneling between the home agent and the MR(Mobile Router). It extends binding messages of Mobile IPv6 and the data transmission of a mobile network can be achieved by using the MR which is the egress interface of a mobile network. In other words, only the MR is involved in the acquisition of CoA (Care of Address) according to a handover of the mobile network. A MNN (Mobile Network Node) which is connected to the MR can maintain their home network address. The NEMO basic support protocol defines basic procedures to support network mobility of a mobile network, but excludes route optimization, multi-

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 673 – 682, 2005.
© IFIP International Federation for Information Processing 2005

homing, and other issues. These issues are examined in the extended network mobility support part. In particular, the basic support protocol has a serious problem called dog-leg problem; that is, all traffic to or from the MNN of the nested mobile network passes through the HAs (Home Agents) of all preceding mobile networks. To solve this problem, various methodologies have been proposed [5], [6], [7]. However, these ideas are still inefficient for real time data transmission and they remain as the optimal route because they are based on bi-directional tunneling between the root-MR and HA of the nested mobile network. Furthermore, direct message exchange between the MNNs within the same root-MR, which is called intra-domain communication, is not supported and the root-MR can experience a very heavy load because it must maintain full paths for all nested MRs. If there are frequent movements of mobile networks, and intra-domain communication is large, these methodologies are very inefficient. Thus, most previous methods have limitations with regard to signaling overhead, concentrated traffic and load in the root-MR, and packet-header overhead due to multiple encapsulations.

Accordingly, this paper proposes a new route optimization methodology for the efficient support of network mobility based on unidirectional tunneling between the HA of a nested mobile network and the root-MR, and the use of a tree-based intra-domain routing mechanism. The use of unidirectional tunneling facilitates more optimized route construction. When using tree-based routing for intra-domain communication and binding procedures, a hierarchical mobile network routing is more efficient and faster than previous tunneling mechanism for signaling and data transmission.

2 Previous Works

The IETF NEMO working group has defined a basic protocol operation to support the network mobility of a mobile network based on Mobile IPv6. There have already been several Internet drafts related to the goals and requirements, terminology, and basic support protocol for network mobility. Network mobility is essentially defined as Nemo basic support and Nemo extended support, where the purpose of Nemo basic support is to preserve session continuity in a mobile network, while Nemo extended support provides more optimal routing for a nested mobile network [4]. The goal of the Nemo basic support protocol is to support network mobility and backward compatibility by extending Mobile IPv6. As such, its definition of the MR extends the MN of Mobile IPv6, where the MR performs internal routing and external data transmission for an MNN, which moves with the MR. Data transmission between the MNN and the CN is performed using bi-directional tunneling between the HA and the MR. All traffic passes through the HA, and IPSec is used for secure signaling between the MR and the HA [4].

In a mobile network on a visited link, a bi-directional tunnel is created between the HA and the MR for data transmission. Thus, data transmission by a mobile network is achieved using the MR, which is the egress interface of the mobile network. In other words, since only the MR is involved in the acquisition of a CoA in a mobile network handover, an MNN behind the MR can maintain its home network address. For the construction of a bi-directional tunnel, the basic support protocol extends the binding

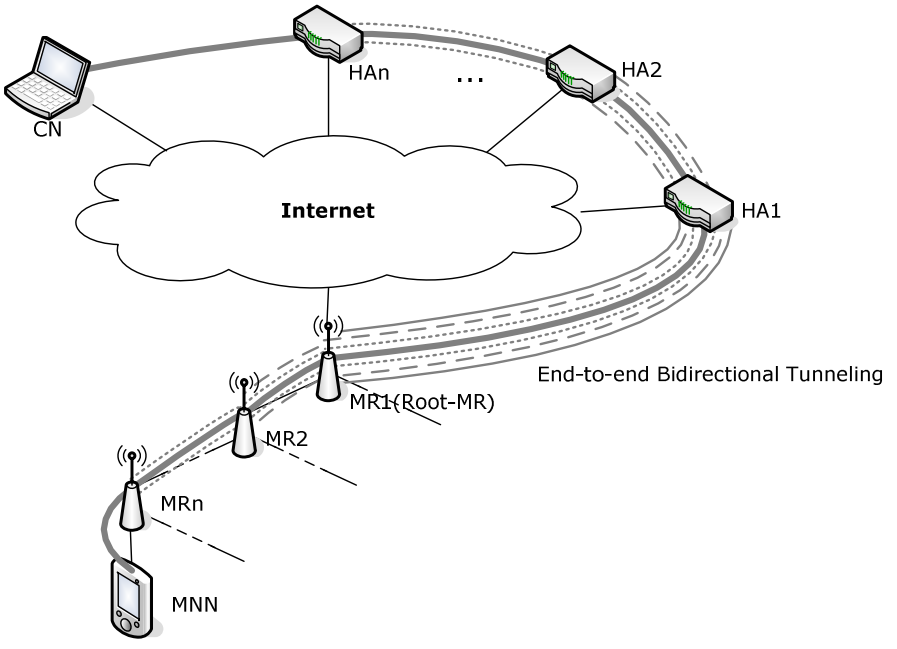


Fig. 1. Dog-leg problem in Nemo basic support protocol

message of Mobile IPv6. The extended BU(Binding Update) message contains a network prefix instead of a home address, and the egress interface address of the MR as the CoA. Thus, by using these extensions, network mobility can be supported without changing the addresses of the MNNs in the mobile network.

The Nemo basic support protocol defines the minimal procedures and extensions required to support network mobility, as such it does not cover route optimization, multi-homing, and so on. Although these issues are being investigated under Nemo extended support, the work has not yet been completed. In the basic support protocol, the tunnel of a nested mobile network is constructed through all preceding mobile network tunnels, and all the traffic of the nested mobile network passes through the HAs of all preceding mobile networks, thereby causing a serious problem called the 'dog-leg problem'. Fig. 1 shows an example of the dog-leg problem in the basic support protocol.

Recently, a lot of research has focused on solving the dog-leg problem, including route optimization of the basic support protocol [5], [7], [9]. Several previous studies on route optimization have used bi-directional tunneling between the HA of the nested mobile network and the root-MR, where two bi-directional tunnels are made between the HA of the nested mobile network and the root-MR, and between the root-MR and the MR of the nested mobile network. By using direct tunneling between the HA of the nested mobile network and the root-MR, the dog-leg problem is solved. Yet, an extended RA(Router Advertisement) message that includes an address of the

root-MR egress interface is required to discover and notify the root-MR address to the nested MRs.

Most of previous works in route optimization have used bidirectional tunneling. The methods using bidirectional tunneling has some drawbacks for real-time data transmission. Since these methods have more complex signaling than the basic support protocol. Because if the root-MR moves along with the nested mobile network, the nested mobile network must re-establish the tunnel since the root-MR address was changed. In addition, these methods do not support intra-domain communication, because all traffic passes through either the HA of the nested mobile network or at least the root-MR. The root-MR must maintain full paths for the MRs of all nested mobile networks. If there are frequent movements of mobile networks, these methods become very inefficient due to the large signaling overhead, large data transmission delay, traffic concentration in the root-MR, and packet header overhead resulting from multiple encapsulations. Therefore, a new route optimization methodology is needed to support efficient signaling and optimized routes.

3 Hierarchical Mobile Network Routing Scheme

3.1 Basic Operation of HMNR

To solve the problems as stated above, this paper proposes a HMNR(Hierarchical Mobile Network Routing) scheme consisting of intra-Nemo routing and extra-Nemo tunneling, where tree-based routing is used for intra-domain data transmission and signaling, while unidirectional tunneling is used for data transmission to or from the external network. Fig. 2 shows the operation of the HMNR scheme for route optimization. As shown in Fig. 2, an MNN behind the MR cannot receive data from the CN directly since a mobile network uses the network prefix of the home network. However, data from an MNN can be directly transmitted to the CN using a normal routing scheme. In other words, although a tunnel is required for data transmission from the HA to the MR, the other tunnel from the MR to the HA does not need to be established to optimize the route. As a result, the MNN can communicate with the CN using only a unidirectional tunnel from the HA to the MR. Similarly, route optimization for a nested mobile network can be achieved using unidirectional tunneling from the HA of the nested mobile network to the root-MR by binding the network prefix to the CoA of the root-MR. Only the root-MR can perform decapsulation of the packets from the CN.

The direct tunneling between the HA of the nested mobile network and the root-MR requires the additional handover procedure because the binding address of the mobile network needs to be changed for the root-MR movement along with the nested mobile network. Thus, the handovers are divided into three cases: inter-Nemo, intra-Nemo and root-MR handover. In the vehicular environments, the root-MR handover frequently occurs and it accompanies with the mass signaling for all nested mobile networks, and the root-MR suffers from the large processing load, bottleneck and the service discontinuity.

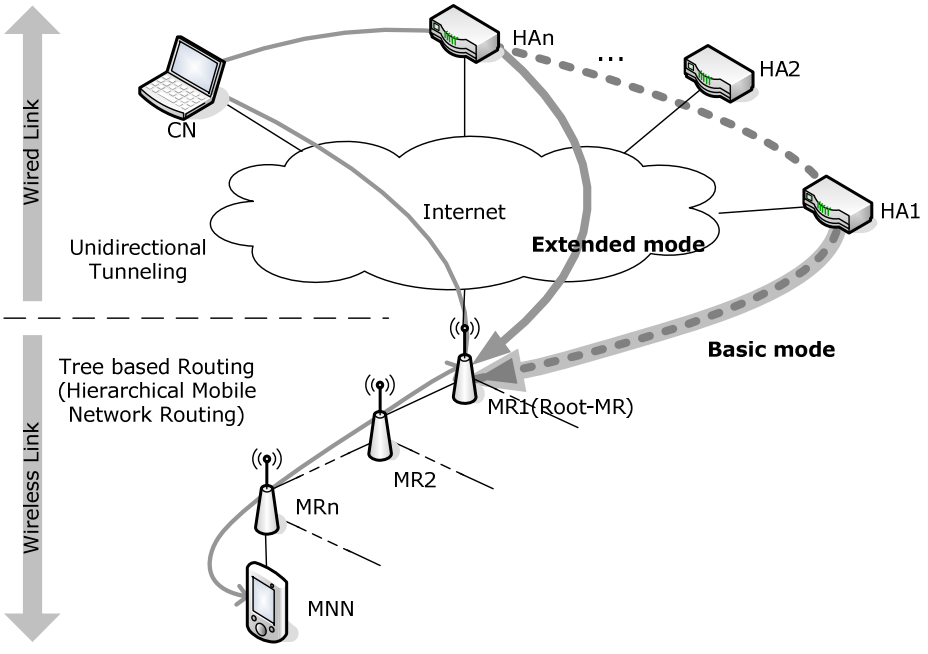


Fig. 2. HMNR Scheme for Route Optimization

To satisfy the requirements, this paper proposes the HMNR scheme with two operating modes for the signaling efficiency and route optimization respectively: the basic mode and the extended mode. In the extended mode, the route optimization is performed by using the direct tunneling from the HA to the root-MR. On the other hands, the MR in the basic mode binds its network prefix to the HoA of the root-MR, thus the data from the CN to the MNN passes through both HAs of the MNN and of the root-MR. By using this mechanism, the MR is independent of the root-MR handover. Moreover, the route is more optimal than the one of the basic support protocol with same signaling complexity because the HMNR scheme has only two intermediate HAs regardless of the nested level of the mobile network. Each operating mode is dynamically switched by sending the BU message including the HoA or CoA of the root-MR for the binding address. Thus the MR decides the basic mode for the vehicular environment with low data traffic, and it can switch into the extended modes if the data traffic or inter-Nemo handover is increased.

3.2 Routing and Handover Procedures

Mobile networks are composed of a tree topology from the root-MR to the MRs of each nested mobile network, where only the root-MR has an egress interface for transmitting data to or from the external network. Thus, a new routing mechanism based on a tree topology is needed where the root-MR is the root of the tree and the nested MRs are the tree nodes. In a tree-based routing scheme, each MR contains a

parent-MR address as a default route entry and maintains a routing table that consists of a mobile network prefix and next hop address pairs for each nested mobile network. In this case, the traffic from the root-MR to the CN is transmitted using the default route entry of the root-MR, that is, an AR(Access Router) instead of tunneling. The routing process is completed by updating the routing entry, which consists of a mobile network prefix and new MR's CoA pair, to the parent-MR, then the parent-MR updates its routing entry and resends the RU message to its parent-MR recursively. At this time, the RU message from the parent-MR has a new next hop address as the CoA of the parent-MR. If an RU message reaches the root-MR or a crossover MR that contains the same routing entry in the RU message, the routing update procedure is completed. The proposed HMNR scheme can also support intra-domain data communication without passing through the HA, because the MR maintains the routing information while providing a routing for transmitting data from MNNs. Traffic can also be transmitted to its destination without passing through the root-MR, if a crossover MR exists which contains a routing entry to the destination.

To support the binding and routing of HMNR, an RA(Router Advertisement) message extension is needed to discover the root-MR and advertise its information. The HMNR also requires a Root-MR Option, which contains the CoA and HoA of the root-MR, and an RA message with the Root-MR Option is used to advertise the addresses of the root-MR.

When a mobile network detects movement, the MR sends an RS(Router Solicitation) message to acquire a network prefix for the foreign network. If an RS message is received, the parent-MR or access router then responds to the MR with an RA message and Root-MR Option. As such, the handover procedures can be decided using an RA message. Fig. 3 shows the RA message handling procedures for supporting an inter-domain and intra-domain handover.

An inter-domain handover includes two cases where the mobile network obtains a new root-MR address: inter-Nemo handover or root-MR handover.

In the former case (*inter-Nemo handover*),

(1) The MR obtains a new CoA
(2) The MR processes a binding procedure between the HA and the root-MR using the address for the root-MR obtained by exchanging RS and extended RA messages with a Root-MR Option.
(3a) The MR sends an RU message with its own network prefix and CoA to the parent-MR for intra-Nemo data transmission.
(4a) Finally, the MR advertises the root-MR address to the nested mobile network by using an RA message with a Root-MR Option.
(3b) Conversely, if the MR receives an RA message without a root-MR address option, this means that the MR is directly connected to the AR, in which case, the MR does not perform a routing procedure, but rather sets itself up as the root-MR
(4b) The MR advertises an RA message with its CoA to the nested mobile network.

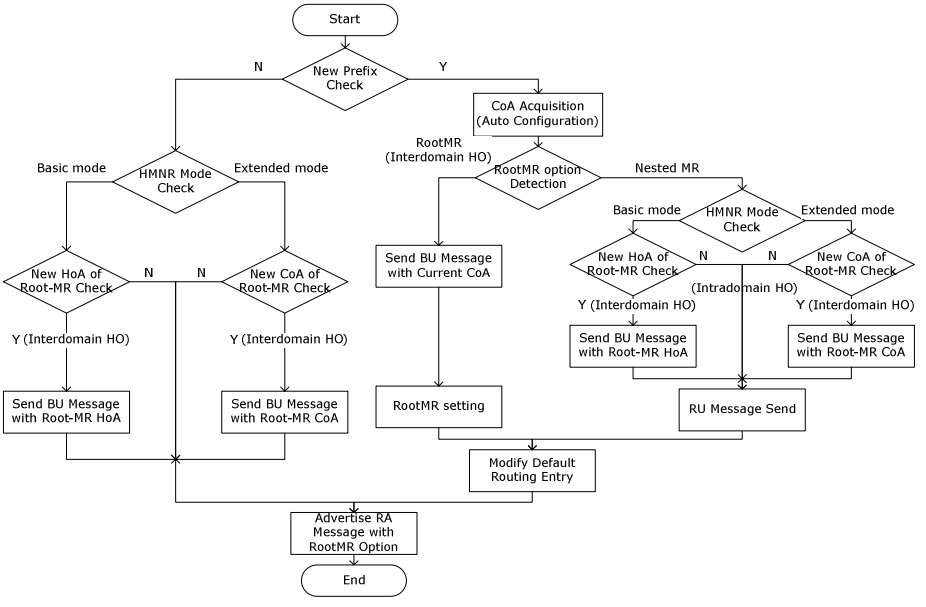


Fig. 3. RA Message Handling Procedures for supporting Inter-domain and Intra-domain Handover

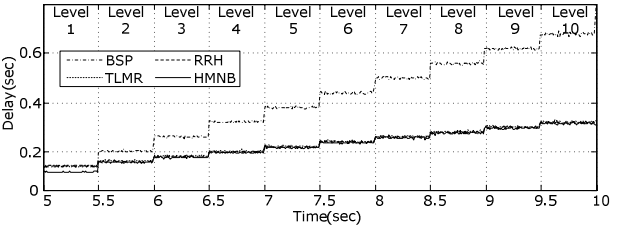
In the latter case (*Root-MR handover*),

(1) The preceding MR obtains a new CoA
(2) The preceding MR performs a handover procedure similar to the former case
(3) Then it advertises a new root-MR address to the nested mobile network. If a mobile network receives an RA message with a new root-MR address
(4) If the MR uses the HMNR extended mode then, the MR reestablishes the unidirectional tunnel between the HA and the new root-MR CoA
(5) Finally, the MR advertises the root-MR address to the nested mobile network.

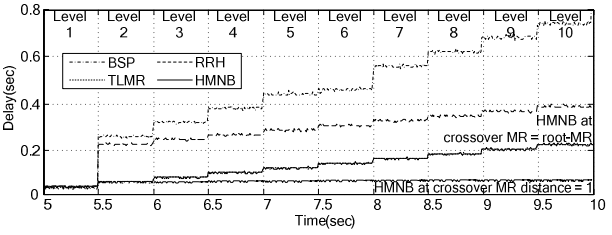
Meanwhile, an intra-Nemo handover occurs when a mobile network moves within the root-Nemo domain, that is, the root-MR address is not changed. In this case, the MR does not need to update the binding to the HA and only performs a routing procedure.

4 Performance Analysis

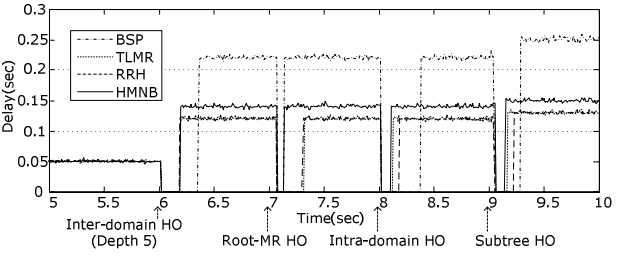
We evaluated the performance of the proposed hierarchical mobile network binding scheme by using discrete event simulation. All HAs are assumed to have the same wired link with a 10ms delay, and the mobile nodes are simulated on the following network mobility support protocols: Basic support protocol (BSP)[2], the route optimization method using bi-directional tunneling (TLMR)[4], Reverse Routing Header (RRH) scheme, and the proposed HMNB scheme.



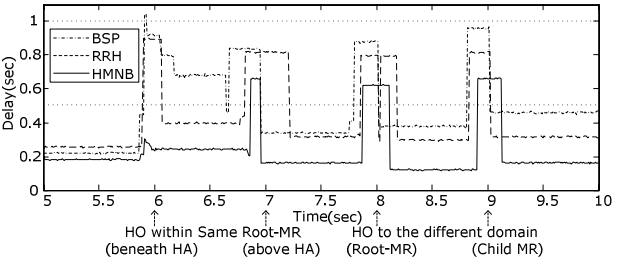
(a) Inter-domain Data Transmission Time vs. MR Level



(b) Intra-domain Data Transmission Time vs. MR Level



(c) Data Transmission Time for CBR HO Scenario



(d) Data Transmission Time for Mobile HA

Fig. 4. The performance analysis of the HMNB scheme

Fig. 4 (a) and (b) show the performance results of the inter-domain and intra-domain data transmission times for the TCP traffic, in accordance with the increase of the depth of the destination MR. In the inter-domain communication environment, the data transmission delays for the route optimization methods i.e. TLMR, RRH and

HMNB schemes were smaller than those of the BSP. In the intra-domain communication environment, the data transmission delays in the RRH scheme are larger than the HMNB scheme because the traffic always passes through the HA. Moreover, the intra-domain communication times of the HMNB scheme show that the delays become smaller than other schemes, if the crossover MR is located below the root-MR. Fig. 4 (c) shows the data transmission delays and the service discontinuity times for the CBR(100 KBps) traffic with the various handover scenarios. In the case of the CBR handover scenario, the delays of the HMNB scheme are larger than those of the RRH and TLMR schemes because it uses two intermediate HAs. The service discontinuity time of HMNB scheme is, however, the smallest in the root-MR handover scenario. At the time of the root-MR handover, the HMNB and BSP schemes can accomplish the handover procedure with just the root-MR, so the discontinuity times of these schemes are the smallest. In the case of the BSP scheme, the nested depth of the MR is deeper, so the service discontinuity time becomes larger than other handover scenarios. Thus, the TLMR and RRH schemes may thus not be suitable for a vehicular environment where the root-MR mobility is usually very high. Fig. 4 (d) shows the performance results of the data transmission times for the mobile-HA scenario, according to the various handover cases. The TLMR scheme does not define any method that supports the mobile-HA. The HMNB scheme has the smallest delay time because the mobile-HA can forward the received binding message to its own HA, so that the traffic does not pass through any additional HAs. In conclusion, the HMNB scheme has minimal signaling complexity and supports an efficient route optimization mechanism for the nested mobile network.

5 Conclusion

In the current study, we have investigated the limitations of current and previous approaches to network mobility management. We have then proposed a new approach for the efficient route optimization in a mobile network, which is called a hierarchical mobile network routing scheme. The proposed scheme uses tree-based routing for intra-domain data transmission and signaling, and unidirectional tunneling for data transmission to or from the external network. It can provide a more optimized solution for route construction and faster signaling, and can also provide intra-domain handover and communication. In addition to these advantages, it can also support micro-mobility without any additional extension.

In summary, the proposed hierarchical mobile network routing eliminates many problems in route optimization such as large signaling delay and the lack of intra-domain communication which are drawbacks of most previous approaches to the mobility management of network groups. We have compared the characteristics of a hierarchical mobile network routing scheme with the NEMO basic support protocol and previous route optimization methods. A hierarchical mobile network routing scheme can be adopted in various mobile environments to efficiently support network mobility such as WPAN, ubiquitous computing, and VNE.

References

1. Hong-Yon Lach, Christophe Janneteau, and Alexandru Petrescu, ”Network Mobility in Beyond-3G Systems”, IEEE Communication Magazine, July 2003
2. Thierry Ernst, ”Network Mobility Support Goals and Requirements”, Internet-Draft, IETF, May 2003
3. Thierry Ernst, and Hong-Yon Lach, ”Network Mobility Support Terminology”, Internet-Draft, IETF, May 2003
4. Vijay Devarapalli, Ryuji Wakikawa, Alexandru Petrescu, and Pascal Thubert, ”NEMO Basic Support Protocol”, Internet-Draft, IETF, December 2003
5. P. Thubert, M. Molteni, and C. Ng, ”Taxonomy of Route Optimization models in the Nemo Context”, Internet-Draft, IETF, June 2003
6. H.S. Kang, K.C. Kim, S.Y. Han, K.J. Lee and J.S. Park, ”Route Optimization for Mobile Network by Using Bi-directional Between HA and TLMR”, Internet-Draft, IETF, June 2003
7. H. Ohnishi, K. Sakitani, and Y. Takagi, ”HMIP based Route Optimization method in a mobile network”, Internet-Draft, IETF, October 2003
8. Campbell, A.T., Gomez, J., Sanghyo Kim, Chieh-Yih Wan, Turanyi, Z.R., and Valko, A.G., ”Comparison of IP micromobility protocols”, IEEE Wireless Communications, Volume 9, Issue 1, February 2002
9. Hyunsik Kang, Keecheon Kim, Sunyoung Han, Kyeong-Jin Lee and Jung-Soo Park, “Route Optimization for Mobile Network by Using Bi-directional Between Home Agent and Top Level Mobile Router”, Internet-Draft, IETF, June 2003

QoS Routing with Link Stability in Mobile Ad Hoc Networks*

Jui-Ming Chen, Shih-Pang Ho, Yen-Cheng Lin, and Li-Der Chou

Department of Computer Science and Information Engineering,
National Central University,
No.300, Jhongda Rd, Jhongli City, Taoyuan County 32001, Taiwan(R.O.C)
Tel: +886-3-422-7151 ext 57968, Fax: +886-3-422-2681
cld@csie.ncu.edu.tw

Abstract. In this paper, in accordance with requirements of different users and supplying effective usage of limited network resources, we propose a stable QoS routing mechanism to determine a guaranteed route suited for mobile ad hoc wireless networks. The manner exploits the received signal strength (RSS) techniques to estimate the distance and the signal change of the velocity to evaluate the breakaway. To ensure the QoS it chooses a steady path from the source to the destination and tries to reserve the bandwidth. Ultimately, it is clear to find that the performance never decrease even the growth of the overhead and the movement of users via the simulated by ns-2.

1 Introduction

Mobile Ad Hoc Wireless Networks (MANET), also called the Ad hoc network, is lots of moving nodes (mobile hosts) communicating with their adjacent mobile node by radio wave. Every node can contact each other without existence infrastructural network. In the Ad hoc network, it differs from cellular wireless networks that need base stations to deliver and receive the packets. Each node plays the role as a router. When one of them wants to deliver packets to destination out of its coverage, intermediate nodes will forward this packet to the next node till the destination node receive it.

In traditional cellular wireless networks, generally we need to establish base stations in advance. Fixed nodes far and near connect to the backbone and become a wireless network environment. In this network the customer who wants to communicate with another must locate in the base station coverage. If user moved out of base station's service scope, he can't take the communication. Consequently, we need to establish enough base stations to achieve the objective. Ad hoc networks do not demand fixed

* This research was supported in part by National Science Council of the Republic of China under the grant NSC 93-2219-E-008-002, and by Computer and Communications Research Labs, Industrial Technology Research Institute under the grant T1-94081-12.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 683 – 692, 2005.
© IFIP International Federation for Information Processing 2005

network infrastructures and centralized management mechanisms, as well as can be built anytime, anywhere rapidly. Ad hoc networks also have the feature of self-creating, self-organization and elf-management as well as deploy and remove network easily. Ad hoc network has above advantages. However, the Ad hoc network environment has the following restricts [1], including of Network topology instable, Limited energy constrained and Limited network bandwidth-constrained

In addition, routing protocol is a very important subject at the Ad hoc networks realm. In general, routing protocols categorize into two types: Table-Driven [2] and Demand-Driven [3]. Though, they both need to have a certain cost to look for and maintain the path. To search a stalest path that has a high throughput is more important than just shortest one. How to find out available, stable and accurate paths and efficient dissipation bandwidth and power energy is an important research at Ad hoc networks [4] [5].

In recent years, the multimedia service makes use popularly to wireless communication. Seeking an available path is not enough to satisfy with customer requests. QoS is also the trend of the future communication [6]. In this paper, in according to the characteristics of Ad hoc network, we propose a routing mechanism that can provide QoS. We exploit the signal strength of two nodes to evaluate a stable path and try to diminish the control signal overhead.

Proactive RTMAC (PRTMAC)[7] is a cross layer framework, with an on-demand QoS extension of Dynamic Source Routing (DSR)[8] routing protocol at the network layer and real-time Medium Access control (RTMAC)[9]. It is designed to provide enhanced real-time traffic support and service differentiation to highly mobile ad hoc wireless networks [10]. It is one of Reservation-based QoS solutions and solves the breakaways and clash. Unfortunately, it must combine with particular MAC layer. Actually, there are some difficult in practice.

2 System Model and Assumption

A. Ad hoc Network Module

In this paper, we model the Ad hoc network system as a graph G = (V, E). The V is a set of mobile nodes in Ad hoc network, and each node has a unique identifier code. The E is all set of arbitrary couple of nodes i and j, and they can communicate by the radio wave. Since the nodes have mobility, V and E will change dynamically. The other relations with fundamental assumptions in Ad hoc network are described as follows:

1.We assume that there is already a MAC protocol which can hold and resolve the problem of media contention and resource reservation effectively.

2.The signal decay and the background noises caused by transmission medium and geography factor are ignored.

3.We assume each mobile node knows others in its coverage, as well as can monitor the signal strength variation of neighbor node anytime. Only the forwarding nodes will receive the messages from broadcasting and the other nodes will drop those.

4.We assume the moving direction and users' speed are random and not influenced by another user. Its average speed maintenance time is T_s. The arrival procedure of user is Poisson Distribution and its average arrive rate is λ (requests/second). The keep time of user call as well as Exponential distribution and its average is T_d and let $\mu = T_d^{-1}$.

Every couple of source and destination node is random selected. In order to evaluate the influence of system by different policies, this paper defines the following reference index:

(1). Routing control overhead ($T_{overhead}$): Assume the normal number of packets is N_c, all the number of packets transmitted is N_{all}, and we define the formula.

$T_{overhead} = N_c / N_{all}$.

(2). The transmission success rate of system ($P_{success}$): the numbers of packets that was received successfully of all network is $N_{receive}$, the numbers of real packets was sent is N_{send}, we define: $P_{throughout} = N_{receive} / N_{send}$

(3). $l_{i,j}$: The link between node i and node j that can transmit message again.

(4). T_{adv}: the period of time that each node broadcasts its status packet.

(5). $BW(i, j)$: the maximum available bandwidth of path from i to j, $P_{i,j}$.

(6). $S_{i,j}(t)$: the node i receive the signal strength from node j at time t, the unit of $S_{i,j}(t)$ is dB.

(7). $T_{i,j}^{predict}$: prediction of the invalidity time of link l_{ij}.

(8). $Path_{s,d}(x)$: if the x th path from node i to node d will pass through nodes (s, x, y,…,z ,w ,d) then we define path is $Path_{s,d}(x) = \{l_{s,x} \cup l_{x,y} \cup \ldots l_{z,w} \cup l_{w,d}\}$

(9). $Path_{s,d}^{predict}(x)$: Prediction of the invalidity time of path $Path_{s,d}(x)$ is

$Path_{s,d}^{predict}(x) = min(T_{s,j}^{predict}, \cdots, T_{w,d}^{predict})$

(10). $RPath_{s,d}$:if we have k path that fit QoS requirement from node i to node d, the most stable routing path is. $RPath_{s,d} = \max(Path_{s,d}^{predict}(x) \mid \forall x = 1,2,\cdots,k)$

(11). We have two different traffic types (QoS and Best Effort) in our system; the ratio of QoS is f.

We can find if the signal strength in distance r_1 is $S(r_1)$, the signal strength in distance r_2 is $S(r_2)$[11], as equation 1:

$$S(r_2) = \frac{r_1^2}{r_2^2}(1 - \rho_d(h_*(r_2)r_2) \times \Delta r) \times s(r_1) \tag{1}$$

In this paper, we assume the notation $\rho_d = 0$. If number of photons sent was N_p, the signal strength in the distance r from source is S, $S = N_p / A_r$. Then the signal strength in distance $2r$ will be $N_p / 4A_r = S/4$; in distance $3r$ will be $S/9$, the rest can be deduced accordingly.

B. The Guarantee Index of Quality of Service

In QoS routing protocol, there are many parameters of QoS guarantee [12][13], such as residual bandwidth, residual buffer space, packet loss probability, delay time and delay jitter, etc. In this paper, we refer to the proposal of Z. Wang [13], only take account of bandwidth to evaluate the efficiency of QoS service. In this paper, we support bandwidth guarantee for QoS type traffic, and we can use some dynamic adaptive technologies to keep the Quality of service.

$$BW_r(s,d) = \min\left(BW_r(s,x),\ldots,BW_r(w,d)\right) \tag{2}$$

3 Proposal of QoS Routing Policy

A. The Prediction Method of Stable Path

In Ad hoc network, we can calculate the variation of signal strength from neighbor link $l_{i,j}$ and predict the broken time with regular beacon signal. In Fig. 1, the signal strength of node *j* at time *t-Δt, t,* is $S_{i,j}(t-\Delta t)$, $S_{i,j}(t)$.Then we use equation 1, and we will derive the answer:

$$\because r_{i,j}(t) = R_t\sqrt{\frac{S_t}{S_{i,j}(t)}},\ r_{i,j}(t-\Delta t) = R_t\sqrt{\frac{S_t}{S_{i,j}(t-\Delta t)}}$$

$$\therefore v_{i,j}(t) = \frac{r_{i,j}(t) - r_{i,j}(t-\Delta t)}{\Delta t} = \frac{\sqrt{S_t}}{\Delta t}R_t\sqrt{\frac{1}{S_{i,j}(t)} - \frac{1}{S_{i,j}(t-\Delta t)}} \tag{3}$$

In Fig. 2, we can detect the same variation rate of signal strength. They have same variation rate but their moving ability are not equal in practice. We define the average variation rate of velocity at *t* time $Avg\left(v_{i,j}(t-\Delta t)\right)$. We divided the average variation rate of velocity into four cases:case1 indicate node *j* approach node *i*, case2 is that node *j* go away from node *i*, case3 is that node *j* start to approach node *i*, case4 shows node *j* leave out the range of node *I*, as in TABLE 1. So we can define a relation equation:

$$Avg\left(v_{i,j}(t)\right) = \begin{cases} \dfrac{Avg\left(v_{i,j}(t-\Delta t)\right) + v_{i,j}(t)}{v_{i,j}(t)}, & \text{Case 1 \& 4} \\ , & \text{Case 2 \& 3} \end{cases} \tag{4}$$

$$T_{i,j}^{predict} = \begin{cases} \left\lfloor \dfrac{R_t - r_{i,j}(t)}{v_{i,j}(t)} \right\rfloor, & \text{if } avg\left(v_{i,j}(t)\right) > 0 \\ T_s, & \text{if } avg\left(v_{i,j}(t)\right) = 0 \\ \left\lfloor \dfrac{R_t + r_{i,j}(t)}{v_{i,j}(t)} \right\rfloor, & \text{if } avg\left(v_{i,j}(t)\right) < 0 \end{cases} \tag{5}$$

Using equation 4, we can derive the relation equation 5 of $T_{i,j}^{predict}$. Using equation 5, we can calculate the predictive lifetime of path, equation 6.

$$Path_{s,d}^{predict}(k) = \min\left(T_{s,l}^{predict}, \cdots, T_{w,d}^{predict}\right) \tag{6}$$

Table 1. $Avg\left(V_{i,j}(t-\Delta t)\right)$ and $V_{i,j}(t)$

Case	$Avg\left(v_{i,j}(t-\Delta t)\right)$	$v_{i,j}(t)$	Status
1	-	-	Keep approahing
2	-	+	Start leaving
3	+	-	Start approaching
4	+	+	Keep leaving

B. The Method of Stable Path and QoS Routing

The method we propose that each node have to maintain two tables, one is Signal Affinity Table (SAT), the other is Zone Route Table (ZRT). Each node will broadcast a routing information packet periodically with a time period T_{beacon}, when a node receives the packet from its neighbor nodes then it records it in its SAT. We can use equation 2.

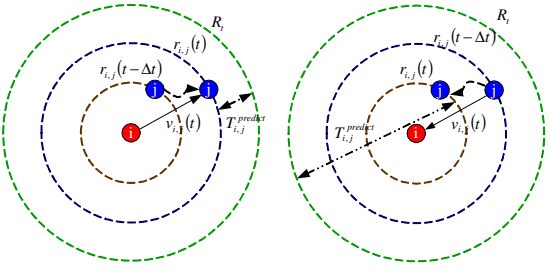


Fig. 1. In the condition of uniform motion, the signal strength of node j per t times

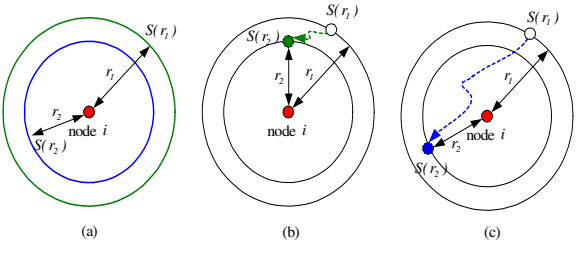


Fig. 2. Different conditions but have the same variation rate of signal strength

Possession of the BW and the prediction of invalidity time of path is $Path_{s,d}^{predict}(x)$ .ZRT will record the information of hops, next node, sequence number, bandwidth, and signal affinity for prediction time.

When a source node wants to transmit the data to destination, it will find its ZRP first. If the destination did not exist in its table, it will broadcast the request to the far node in its table. The node receive the request packet, they will repeat the step before the destination in they ZRT. When the destination receives the route request packet, it will wait a back-off time; then select the path which has the longest predictive time; then we use this path to transmit the data. Show as equation 7.

$$RPath_{s,d} = \max\{Path_{s,d}(k)\} \tag{7}$$

If has more than one longest path exist, we select the path which has the least number of the hops. When the predictive time of path is equal or less than 1 sec, we will pre-route before the path is failure.If the load of network is heavy, the efficiency of DSDV is better than DSR; otherwise DSR is better than DSDV [14]. In this paper, we attempt to find a QoS routing path using DSR. The destination received the request packet, he will find the path which available bandwidth conformed to the request and it has the longest predictive longest life time. If no path conforms to the request, the destination will decide one with longest predictive time and randomly drop some best effort traffic to reserve the bandwidth whole path.

4 Performance Evaluation

A. Simulation Environment

The *ns-2* [15] simulator was used in the simulation study, as well as can be exploited for academic research. In the simulation, the environment parameters of the system was refer to the reference [14], Inter-Frame Space (IFS) is 10 μs , Short IFS (SIFS) is 10 μs , PCF IFS is 30 μs , DCF IFS is 110 μs , every time slot is 20 μs . If a packet stays in the queue buffer over 20 ns and was not sent yet, it will be dropped. Contention Window, CW, CW_{min}=32 and CW_{max}=1024. The maximum size of interface queue is 64, all nodes communicate with identical, half-duplex wireless radio that are modeled available IEEE 802.11-based of 2Mbps and a nominal transmission radius of 250 meters. T_{beacon} = 1 ms. The mobility model referred to [2].Our simulation modeled a network of 50 mobile hosts placed randomly using uniform distribution within 1000 meter multiplied by 300 meter area. The minimum and the maximum velocity set zero and 20 m/s, respectively, while each time user velocity is determined previously, and such velocity maintains 100 s, T_s.

In our simulation model, the transmission ratio uses CBR traffic sources between the communications of nodes, as well as data packets are 64 or 1024 bytes long. The velocities of transmission are a packet per second, four packets per second and eight packets per second. The user arrival procedure is Poisson distribution, as well as the

average arrival rate is λ . That $\mu = T_d^{-1}$ and user calling duration is exponential distribution. The average duration T_d is 180s.Their are 1000 calling-requests during the simulation.

B. Simulation Results

Fig. 3 shows the influence of transmission success rate and control overhead by the speed of nodes. Because our proposal possesses the ability of prediction invalidity time of path, we can handle it effectively. Fig. 4 shows the influence of transmission success rate, control overhead and control packet ratio by the arrival time 1/λ (seconds per call), and vice versa the arrival rate defines λ.

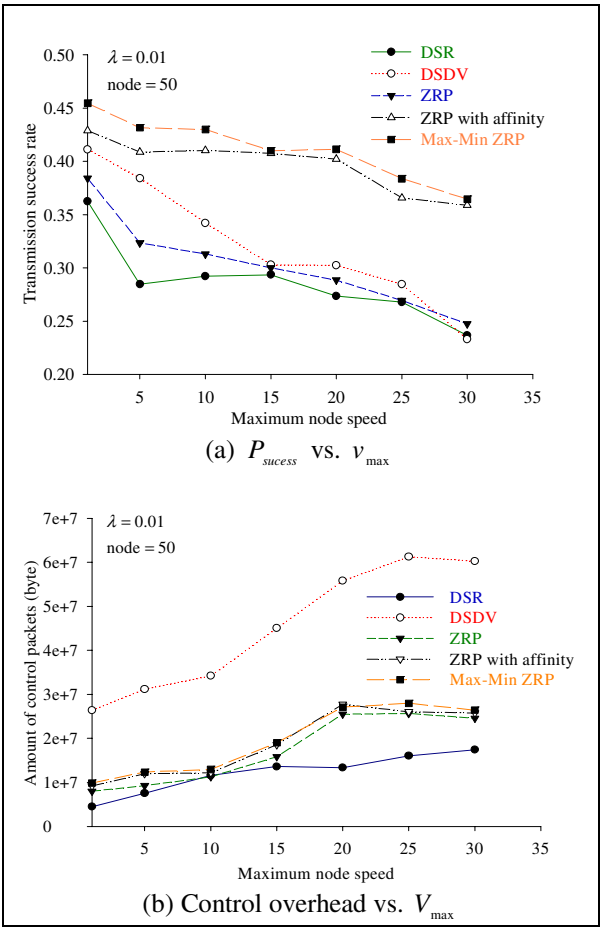


Fig. 3. v_{max} vs. throughput and overhead

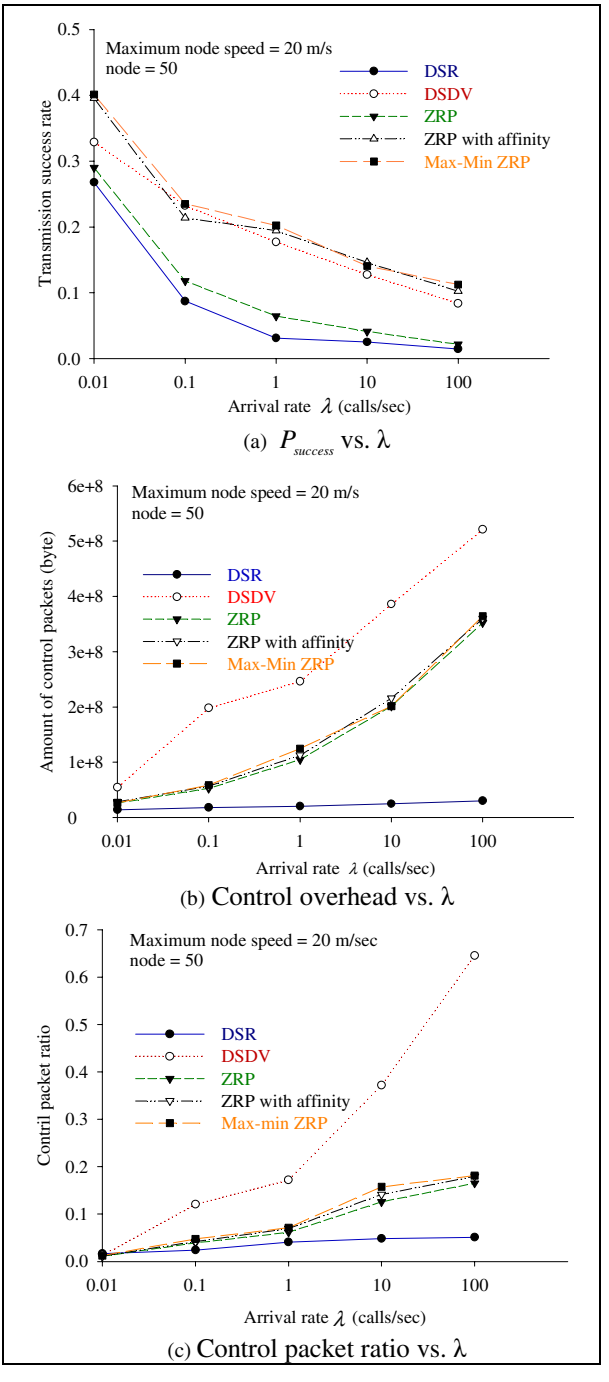


Fig. 4. λv.s. P_{sucess} , control overhead and control packet ratio

We can discover when the arrival rate of user calls increase, the control overhead of DSDV is higher than DSR. Because DSDV must exchange its information with neighbor nodes, so its control overhead will increase, and the ratio of control packet will be higher. The method we propose is based on ZRP. Consequently, the control signal overhead is between those of DSR and DSDV. We simulate the method we proposed with QoS scheme, we divided the zone radius into Z=2. We can discover when the arrival rate is lower, the route discovery latency will be smaller, but when the arrival rate is higher, the route discovery will increase. We can discover the more QoS type traffic that must cause more call blocking rate. If the load of network is very heavy, new calls will be sacrificed; the call blocking probability will be increased. We find out the loss rate is less than all Best Effort traffic with our proposal and just only 30~40% miss rate with our predictive policy. Similarly, the transmission success rate is also higher than ZRP even without pre-route of our method. If the load of network is very heavy, the route discovery latency of network may be long, but they are also much less than 1 sec that is the unit of the predictive time.

5 Conclusion

The Ad hoc network makes it easy to build up a network environment anytime, anywhere. But the node mobility will break down the paths transmitting data and make the system ineffective. So we propose a method to prediction the broken time of a path and can find a most stable path. According to the simulation results, we can discover if without the QoS service, the transmission success rate of the most stable path we select can increase at most 40% of the ZRP, and its control load can in a certain range. When QoS service is taken into consideration, if the load of network is very heavy, the packet loss rate of network may also stays at 60% below, but new calls will be sacrificed, the call blocking probability will increase. The transmission success rate of our method without the function of pre-route is also better than ZRP. So the Stable path is useful.

References

[1] M. Conti and S. Giordano, “Mobile Ad-hoc Networking,” Proceedings of IEEE 34th Annual Hawaii International Conference on System Sciences (HICSS-34), vol. Abstracts, pp. 250-250, Jan. 2001.

[2] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu and J. Jetcheva, “A Performance Comparison of Multihop Wireless Ad Hoc Network Routing Protocols,” *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM’98)*, Dallas, Texas, USA, pp. 85-97, Oct, 1998.

[3] J. Raju and J. J. Garcia-Luna-Aceves, “A Comparison of On-Demand and Table Driven Routing for Ad-Hoc Wireless Networks,” *Proceedings of 2000 IEEE International Conference on Communications (ICC’2000*, New Orleans, Louisiana, USA, vol. 3, pp. 1702-1706, June 2000

[4] J. -H. Ryu, Y. -W. Kim and D. -H. Cho, "A New Routing Scheme Based on the Terminal Mobility in Mobile Ad-Hoc Networks," *Proceedings of 1999 IEEE 50th Vehicular Technology Conference (VTC'99)*, Amsterdam, Netherlands, vol. 2, pp. 1253-1257, Sept. 1999.

[5] K. Paul, S. Bandyopadhyay, A. Mukherjee and D. Saha, "Communication-Aware Mobile Hosts in Ad-hoc Wireless Network," *Proceedings of 1999 IEEE International Conference on Personal Wireless Communication (ICPWC'99)*, Jaipur, India, pp. 83-87, Feb. 1999.

[6] S. Chen and K. Nahrstedt, "Distributed Quality-of-Service Routing in Ad Hoc Networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1488-1505, August 1999.

[7] V. Vishnumurthy, T. Sandeep, B. S. Manoj and C. S. R. Murthy, "A novel out-of-band signaling mechanism for real-times support in tactical ad hoc wireless network," Real-Time and Embedded Technology and Application Symposium, 2004, Proceedings of RTAS 2004, 10th IEEE, pp. 55-63, May 2004.

[8] D.B. Johnson and D.A. Matlz, "Dynamic Source Routing in Ad Hoc Wireless Networks," Mobile Computing, Kluwer Academic Publishers, vol. 353, pp. 153-181, 1996.

[9] B. S. Manoj and C. Siva Ram Murthy, "Realy-time Traffic Support for Ad Hoc Wireless Networks," Proceedings of IEEE ICON 2002, p. 335-340, August 2002.

[10] T. Bheemarjuna Reddy, I. Karthigeyan, B. S. Manoj, and C. Siva Ram Murthy, "Quality of Service Provisioning in Ad Hoc Wireless Networks: A survey of Issues and Solution," Technical Report, Department of Computer Science and Engineering, Indian Institue of Technology, Mardras, India, July 2003.

[11] W. C. Lynch, K. Rahardja, S. Gehring, "An Analysis of Noise Aggregation from Multiple Distributed RF Emitters," *Interval Research Corporation*, Dec. 1998.

[12] S. Chakrabarti and A. Mishra, "QoS Issues in Ad Hoc Wireless Networks," *IEEE Communications Magazine*, vol. 39, no. 2, pp. 142-148, Feb. 2001.

[13] Z. Wang and J. Crowcroft, "Quality-of-Service Routing for Supporting Multimedia Applications," *IEEE Journal on Selected Areas in Communications,* vol. 14, no. 7, pp. 1228-1234, Sept. 1996.

[14] S. R. Das, C. E. Perkins and E. M. Royer, "Performance Comparison of Two On demand Routing Protocols for Ad Hoc Networks," *IEEE INFOCOM'2000*, Tel Aviv, Israel, vol. 1, pp. 3-12, March 2000.

[15] The Network Simulator – ns-2, http://www.isi.edu/nsnam/ns/.

Efficient Cooperative Caching Schemes for Data Access in Mobile Ad Hoc Networks

Cheng-Ru Young[1,2], Ge-Ming Chiu[1], and Fu-Lan Wu[3]

[1] Department of Computer Science and Information Engineer,
National Taiwan University of Science and Technology
{chiu, D8815005}@mail.ntust.edu.tw
[2] Department of Electronic Engineering, Chin Min College of Technology and Commerce
chengru@ms.chinmin.edu.tw
[3] BenQ Corp. Taipei, Taiwan
fulan@totoro.ee.ntust.edu.tw

Abstract. We study cooperative caching technique for supporting data access in ad hoc networks. Two protocols that are based on the notion of zone are proposed. The IXP protocol is push-based in the sense that a mobile node would broadcast an index message to the nodes in its zone to advertise a caching event. A data requester can fetch a needed item from a nearby node if it knows that it has cached the data. The second protocol, called DPIP, is explicitly pull-based with implicit index pushing property. A data requester may broadcast a special request message to the nodes in its zone asking them to help satisfy its demand. However, this is done only if its own caching information does not result in a successful fetch. Performance study shows that the proposed protocols can significantly improve system performance when compared to existing caching schemes.

1 Introduction

Recent advances in wireless communication technology have greatly increased the functionality of mobile information services and have made many mobile computing applications a rlity. It provides users with ability to access information and services through wireless connections that can be retained while the users are moving. A number of novel mobile information services, such as mobile shopping aids in a large shopping mall and financial information distribution to users via mobile phones and palmtop computers, have alreaeady been implemented.

While many infrastructure-based wireless networks have been deployed to support wireless communication for mobile hosts, they suffer from such drawbacks as the need for installing base stations and the potential bottleneck of the base stations. The ad-hoc-based network structure (or MANET) alleviates this problem by allowing mobile nodes to form a dynamic and temporary network without any pre-existing infrastructure. This can be highly useful in some environments. For example, in a large shopping mall there may be an info-station that stores the prices of all goods for que-

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 693 – 702, 2005.
© IFIP International Federation for Information Processing 2005

rying. Due to limited radio range, an info-station itself can only cover a limited geographical area. If shoppers' mobile devices are able to form an ad hoc network, they can have access to the price list even if they are beyond the radio range of the info-station. In such an environment, when a request for a price is forwarded toward the info-station, it is likely that one of the nodes along the path has already cached the requested item. This node can send the data back to the requester so that the access time, the channel bandwidth, and the battery power can be saved.

In light of the above example, we see that caching of data offers significant benefits for applications in ad hoc networks. Cooperative caching, which allows sharing and coordination of cached data among multiple nodes, has been widely used to improve web performance in wired networks [9, 22, 25]. These protocols usually assume fixed network topology and often require high computation and communication overheads. However, in an ad hoc network, the network topology changes frequently, and mobile nodes typically have resource (battery, CPU, and wireless bandwidth) constraints, thus cannot afford high computation or communication overheads. Moreover, ad hoc networks are based on wireless communications that are unreliable in nature. Being able to access data from nearby nodes is important from performance point of view. Hence, there exists a need for new techniques that may be applied to ad hoc networks.

In this paper, we design and evaluate cooperative caching techniques for supporting data access in ad hoc networks. Two protocols that are based on the notion of zone are proposed. In the protocols in-zone broadcasts of small-sized messages are exploited to assist the locating of required data items. The first protocol, called *IXP*, is push-based in the sense that a mobile node broadcasts an index message to the nodes in its zone to advertise a caching event. A data requester can fetch a needed item from a nearby node if it knows the node has cached the item. Otherwise, it issues a request to the data center to ask for it. Any nodes along the path can redirect the request to a nearby node, instead of faraway data center, if it knows the node has cached the item. The second protocol, called *DPIP*, is explicitly pull-based with implicit index pushing property. A data requester may broadcast a special request message to the nodes in its zone asking them to help satisfy its demand. However, this is done only if its own caching information does not result in a successful fetch. *DPIP* allows a wider scope of local caching cooperation without incurring extra communication overhead. In particular, unlike previous techniques [5, 6, 7], *DPIP* exploits the implicit index pushing property in locating data items existing in nearby nodes. In addition, the proposed protocols achieve a greater level of caching cooperation through employing an appropriate cache replacement mechanism. Simulation results show that the proposed protocols may improve system performance in terms of the request success ratio and the data access time.

2 Related Works

To facilitate data access in ad hoc networks, some data replication schemes [11, 12, 13, 24] and caching schemes [18, 23, 26] have been proposed in the literature. Data replication addresses the issue of allocating replicas of data objects to mobile hosts to meet access demands. These techniques normally require a priori knowledge of the operation environment and are vulnerable to node mobility.

Unlike data replication schemes, caching schemes do not rely on distributing data objects beforehand to facilitate data access. In the conventional caching scheme, referred to as *SimpleCache* [26], a data requester always caches the received data. If subsequent requests for the cached data arrive before the cache expires, the node may use the cached copy to serve the requests. In case of a cache miss, it has to get the data from the data center. Getting data from faraway data center will increase the response times for the requests. Recently, a cooperative caching strategy, called *CoCa*, was proposed in [5, 6]. In *CoCa*, mobile hosts share their cache contents with each other to reduce both the number of server requests and the number of access misses. In addition, a group-based cooperative caching scheme, called *GroCoCa*, has been presented in [7], in which a centralized incremental clustering algorithm is adopted by taking node mobility and data access pattern into consideration. *GroCoCa* improves system performance at the cost of extra power consumption. In the *7DS* architecture [19], users cache data and share with their neighbors when experiencing intermittent connectivity to the Internet. However, the above researches focus on the single-hop environment rather than the multi-hop ad hoc networks addressed in this work.

In ad hoc networks, finding the location of a cached copy of a data item is the core of a cooperative caching mechanism. In [17], when an object is requested, the protocol relies on flooding to find the nearest node that has maintained a copy of the object. Using flooding may potentially reduce the response time since the request can be served by a nearby node, instead of the data center faraway. However, flooding can be problematic for network communication. To reduce the overhead, flooding is limited to nodes within k hops from the requester in [17], where k is the number of hops from the requester to the data center. The overhead is still excessive especially when k is large or the network density is high. In [18], flooding is limited by imposing a threshold on route existence probability. Based on the definition of route stability, as a query packet is forwarded by hopping, its route existence probability becomes smaller. By loading the threshold of route probability into the header of a request packet beforehand, the range of cache querying can be limited. However, choosing appropriate threshold for route existence probability is challenging.

Hence options other than flooding are desirable for finding a needed data item in mobile ad hoc networks. In [23], a cooperative caching scheme has been proposed to reduce the communication and energy costs associated with fetching a web object. When a terminal M wants to get a web data W that was not cached locally, M requests W through the base station only if the base station is in the zone of M. Otherwise, M will broadcast a request message for W in its zone. If W is not cached by any of the mobile nodes in the zone, a peer-to-peer communication scheme is realized with the mobile nodes that are known to share interests with M and are at a distance that is less than the one between M and the nearest base station. The communication is based the notion of terminal profile. However, if the data correlation between mobile terminals is small, the effect of terminal profile will be lost. In [26], two caching schemes, called *CacheData* and *CachePath*, had been presented. With *CacheData*, intermediate nodes may cache data to serve future requests. In *CachePath*, a mobile node may cache the path to a nearby data requester while forwarding the data and use the path information to redirect future requests to the nearby caching site. A hybrid protocol *HybridCache* was also proposed, which improves the performance by taking advantages of *CacheData* and *CachePath* methods while avoiding their weakness. In *HybridCache*, when a mobile

node forwards a data item, it caches the data or the path based on some criteria. These criteria include the data item size, and the time-to-leave value of the item. One problem with these methods is that caching information of a node cannot be shared by a data request if the node does not lie on the path between the requester and the data source.

3 Zone-Based Cooperative Caching Schemes

Our research is motivated by ZRP, a zone-based routing protocol [1, 10]. In general, routing protocols for MANETs can be classified into two categories: proactive and reactive. Proactive protocols (e.g., OLSR [8], DSDV [21]) update their routing tables periodically. Reactive protocols (e.g., AODV [20], DSR [16]), on the other hand, do not take any initiative in finding a route to a destination until a routing demand arises, thus a priori reduce the network traffic. ZRP is a hybrid protocol that combines reactive and proactive modes. In ZRP, a zone is associated with each mobile host and consists of all the nodes that are within a given number of hops, called *radius* of the zone, from the host. Each node proactively maintains routing information for the nodes in its zone. In contrast, a reactive protocol is used to reach any node beyond its zone.

We use the notion of zone as ZRP in this research. The basic idea of our scheme is to have each mobile host share its caching contents with those in its zone or beyond without the need for group maintenance. Our design rationale is twofold. As stated previously cooperative caching is possible among neighboring nodes, and zone reflects such notion of vicinity. Second, aided by the underlying routing protocols such as ZRP, a zone can be readily formed and maintained for a mobile host even if the node is on the move. In the following, we present a simple protocol called *Index Push* (or *IXP*) and a more sophisticated one called *Data Pull*/*Index Push* (or *DPIP*) for implementing the zone-based cooperative caching.

3.1 System Model

We consider a MANET where all mobile nodes cooperatively form a dynamic and temporary network without any pre-existing infrastructure. There exists a *data center* that contains a database. Each mobile host may send a request message to the data center for accessing a data item. When a node fetches a data item, it always stores the item in its local cache for future use like conventional caching scheme. We assume that each node has limited cache space, so only a portion of the database can be cached locally. If the cache space of a node is full, the node needs to select a victim data item from its cache for replacement when it wants to cache a new one. To reduce access latency and to ease the load of the data center, an intermediate node on the forwarding path between the requester and the data center can directly deliver the requested data to the requester if it has a copy in its local cache, or redirect the request to some nearby node that it knows has cached the data item.

In our system, we also make the following assumptions:

(1) All data items are of the same size.
(2) For sake of simplicity and standing out the salient features of our proposed schemes we do not consider updates of the data items.

(3) We assume that ZRP is the underlying routing protocol used in the MANET although this is not indispensable for our schemes.

3.2 *Index Push* (*IXP*) Protocol

The idea of *Index Push* is based on having each node share its caching content with those in its zone. To facilitate exposition, we call the neighboring nodes in the zone of a node M the *buddies* of M. If M_1 is a buddy node of M, M is also a buddy of M_1. To this end a node should make its caching content known to its buddies, and likewise its buddies should reveal their contents to the node. One way of arriving at this is for a node that has cached a new data item to advertise such a caching event. *Index Push* (*IXP*) takes this approach. It broadcasts an index message to its buddies whenever it caches a data item. The id of the data item that has been cached is included in the index message. A node may receive multiple index messages from different buddies that are associated with the same data item. Each node maintains an index vector, denoted as IV. An IV has N elements, where N is the number of data items; each element is associated with a distinct data item. Each element in IV has three entries that are used to record the caching information of the corresponding item. Consider the IV of a node M. The first entry associated with data item x is of type binary and is represented by $IV[x].cached$. This entry indicates whether x is cached locally. If the entry is TRUE, it means that x is locally available; otherwise x has to be obtained from some remote site. The second entry, denoted as $IV[x].cachednode$, is used to record a nearby node that has cached x. For sake of saving storage space M only records the last node that has broadcast an index message associated with x. The third entry, represented by $IV[x].count$, contains a count of M's buddies that are known to have cached x since x is cached by M. As described later, this count will be used for cache replacement purpose. Initially, any $IV[x].cached$ is set to FALSE, $IV[x].cachednode$ is set to NULL, and $IV[x].count$ is set to zero.

(a) Data Accessing/Caching

Consider that a node M wants to access a data item x. M first checks its $IV[x].cached$ to see whether the data item has been cached locally. If the entry is FALSE, M proceeds to examine $IV[x].cachednode$ expecting someone in the neighborhood may offer some help. If the entry is NULL, M sends a request message to the data center directly. If the entry is not NULL, M issues a request message to the node, say M_1, if M_1 is still in the zone. Due to node mobility, it is possible that M_1 may no longer stay in M's zone. With ZRP, if M_1 is not inside M's zone M has no routing information about how to reach M_1. To avoid the overhead for searching for a path to M_1, M would send a request message to the data center directly. An intermediate node I on the path to the data center can redirect the message to a buddy node that I knows has cached the data item according to its $IV[x].cachednode$ entry. When M eventually receives x, it caches x. In doing so it may possibly need to discard another cached data item, say y, if its cache is full. M sets its $IV[x].cached$ to TRUE and $IV[y].cached$ to FALSE. Moreover, M will reset its $IV[x].count$ to 0. As described in the next section, this is performed in order to avoid having x be chosen for replacement soon after it is cached. M then broadcasts an index message to its buddies. Included in the index message are ids of the newly cached item x and the replaced data item y. Upon receiving the index message M's buddies update

their *IV[x].cachednode* entries with *M*, increase *IV[x].count* by one and decrease *IV[y].count* by one. The last two operations, i.e. updating *IV[].count* entries, need to be performed by a buddy node only if the corresponding data items are cached by the node. Furthermore, if a buddy has recorded *M* in its *IV[y].cachednode*, it needs to set the entry to NULL because *y* is no longer cached by *M*.

(b) Cache Replacement

If a node *M* accesses a data item when its cache space is full, some cached item must be removed to make room for the new one. In *IXP*, we use *IV[].count* entry for cache replacement. Recall that this entry indicates the number of *M's* buddies that have cached a data item since *M* cached the same item. *IXP* replaces the data item that has the maximum *IV[].count* among all cached ones. Replacing such a data item tends to induce less impact on *M*'s buddies because there are less buddies relying on *M* for fetching the data item when the associated count becomes bigger. In addition, *M* and its buddies tend to have greater chance in finding the replaced data item in their neighborhood than in finding the other cached items. Moreover, doing so can limit caching duplicates. This may also explain why *M* needs to set the *IV[x].count* to 0 when it first caches a data item *x*. At this time, all of *M's* buddies are supposed to have their *IV[x].cachednode* entries point to *M*, hence we do not want to have *x* replaced too soon. Notice that once a data item *x* is chosen for replacement, the values of *IV[x].count* maintained by *M*'s buddies will also be decremented, implying less chance for these buddies to have *x* replaced. This can alleviate the problem of concurrently replacing the same item by all the nodes in the neighborhood.

3.3 *Data Pull/Index Push (DPIP)* Protocol

IXP is essentially push-based in the sense that a caching node "pushes" the caching information to its buddies. Each node has a view of the caching status in its zone only. However, due to mobility of the nodes and some limitations of mobile devices such as transient disconnection, the caching status reflected by *IV* may be obsolete or not up-to-date. For example, suppose that, according to node *M's IV*, none of *M's* buddies caches *x*. If a new node that has cached *x* moves into *M's* zone now, this caching information cannot be captured by *M's IV* with *IXP*. In the following, we propose a more sophisticated protocol called *Data Pull/Index Push* (*DPIP*) to deal with this problem.

Similar to *IXP*, each node maintains the *IV* vector. When a node *M* wants to access a data item *x* that is not cached by itself, it first examines the entry *IV[x].cachednode* to see if some buddy node in the zone has cached *x*. If such buddy node exists, *M* sends a request message to the node asking for a copy of *x* in the same way as *IXP*. However, unlike *IXP*, if *IV[x].cachednode* entry is NULL, *M* broadcasts a special request message *srg* to all of its buddies. The *srg* message carries the ids of the requested data item, *x*, and the data item that will be replaced if the cache space is already full. Upon receiving the *srg* message a buddy node will reply to *M* if either of the following conditions is met: (1) it has cached *x*, (2) it knows some of its buddies has cached *x* (as per its *IV*). To reduce the number of reply messages, only peripheral nodes, i.e. the nodes at the perimeter of *M's* zone, are required to reply under the second condition, but any node for which the first condition is met should reply. In contrast with *IXP*, *DPIP* increases the chance for the data requester *M* to obtain a copy of *x* from nodes in its vi-

cinity. This can be argued as follows. In addition to the fact that *M* can fetch a copy of *x* from its buddies if the first condition is met, it may possibly obtain caching status of the nodes that are beyond its zone but within the zones of its buddies as specified by the second condition. Consequently, the scope of local cooperation is essentially extended by a factor of two, in terms of radius, at the data requester site. In addition, the in-zone broadcast, which initiates the "data pulling" operation, allows *DPIP* to use the latest caching information. From above description, we see that in-zone broadcast of the *srg* message implicitly advertises the fact that the requester node will cache a copy of the requested data item, and it will soon be able to help others in satisfying their demands for the item. In other words, *srg* messages serve two functions: data pulling and index pushing (in implicit manner). Note that when a *srg* message is broadcast, a timer is started. If no reply is received from *M's* buddies before the timer goes off, *M* sends a request message toward the data center. *IV* vectors are updated in the same way as *IXP* at both the requester site and its buddies. However, the update of *IV* is done by the buddies at the time when they receive either a broadcast *srg* message or a direct (unicast) request message from the requester site. Cache replacement is performed in the same way as *IXP*.

4 Simulation Study

In this section we present simulation results for the proposed protocols and compare them to *HybridCache*.

4.1 Simulation Configuration

We have developed a simulation model based on the ns-2 simulator [27] with CMU wireless extension. The simulated network consists of 50 nodes spread randomly in an area of 1500m×320m, similar to the model used in [26]. One node is designated as the data center, and it is fixed at the upper left-hand corner of the area throughout the simulation. For ease of implementation, we use the DSDV [21] as the underlying routing protocol. It is assumed that the wireless transmission range of the nodes is 250 meters and the channel capacity is 2Mbps. A node moves according to the random waypoint model [2], in which each node selects a random destination in the specified area and moves toward the destination with a speed selected randomly from the range (1m/s, 10 m/s). After the node reaches its destination, it pauses for a period of time and then repeats the movement pattern. The pause time is used to represent node mobility. The default pause time is 300 seconds.

The data center contains 3000 data items. Data updates are not considered in the model. Data requests are served on FCFS basis at all nodes. The size of each data item is 1000 bytes, other messages such as request messages and index messages are all assumed to be 20-byte long. Each node generates a sequence of read-only requests. The inter-arrival times of data requests follows exponential distribution with mean 20 seconds. The data items are accessed uniformly. We consider a data request failed if the requested data item is not returned within a given amount of time. This is employed to account for packet loss in the unreliable wireless network. In addition, a data request that takes an excessive amount of response time is, in most cases, abandoned by the client.

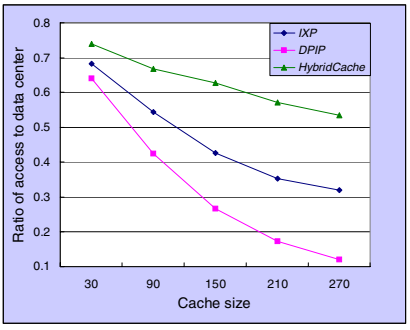


Fig. 1. Request failure ratio vs. cache size

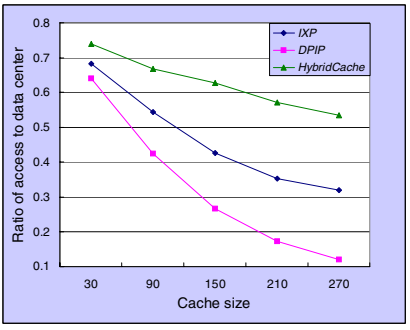


Fig. 2. Ratio of access to data center vs. cache size

To capture the performance of the protocols several metrics are examined in the simulation. The first principal metric is the request failure ratio which gives the ratio of data requests that fail to receive the requested data items. The second metric we measure is the average data access time for a successful data request. The other metric of interest is the ratio of data requests that are served by the data center. Reducing such ratio mitigates the workload of the data center, and better load balance results.

As we are interested in the steady state of the system we allow the simulated network to warm up for 1000 seconds. The simulation results are collected for 4000 seconds. In our simulations the radius of a zone is set to one hop for both *IXP* and *DPIP* protocols.

4.2 Simulation Results

Figure 1 illustrates the request failure ratios for *IXP*, *DPIP* and *HybridCache* with cache size varying from 30 to 270 items. In this simulation the pause time is set to 300 seconds. Apparently, both *IXP* and *DPIP* outperform *HybridCache* by significant margins. In addition, traffic congestion near the data center may cause request failures as well. Note that our schemes, *DPIP* in particular, demonstrate more evident improvement, in comparison with *HybridCache*, as the cache size increases. This result indicates that our protocols offer great capability for exploiting localized cooperative caching.

Figure 2 shows the ratios of the requests messages that are eventually addressed to the data center. Reducing the chances of having data requests satisfied by the data center is essential to the efficiency of the cooperative caching techniques. *DPIP* is least likely to lead the requests to be sent to the data center, followed by *IXP* and then *HybridCache*. A smaller such ratio implies, to some extent, better load balance among the data center and the nodes that have cached the requested data items. Again, *DPIP* is most sensitive to the cache size with respect to this metric.

In Figure 3 we illustrate the average access times of the successful data requests for the same setting as used in Figure 1. Since the access times do not consider the failed data requests, a straightforward comparison of the three protocols using this figure may not be appropriate. From Figure 3 we see that *DPIP* does not offer advantages in terms of the average access time when the cache size is very small. This is mainly due to the fact

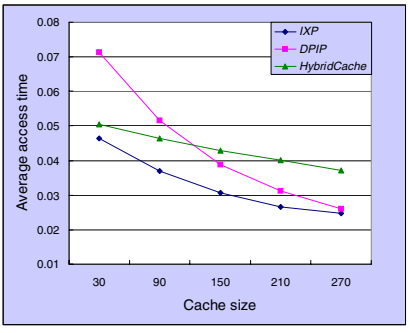


Fig. 3. Average data access time vs. cache size

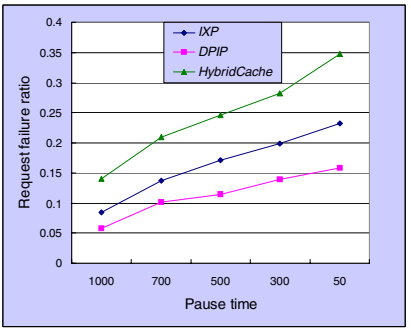


Fig. 4. Request failure ratio vs. pause time

that *srg* broadcasts in *DPIP* are not effective enough to compensate for the timeout intervals experienced by the data requesters when no cache hit among their buddies results. In this situation, both *IXP* and *HybridCache.*

To evaluate the effect of node mobility on the performance of the protocols we have performed simulation with different pause times. The result is shown in Figure 4, in which the request failure ratios are plotted against the pause times. Note that as the pause time decreases the node mobility becomes higher. All three protocols are affected when node mobility increases. *DPIP* is least sensitive to node mobility because it provides a comparatively wide scope of caching cooperation for neighboring mobile nodes. *HybridCache* is the most sensitive with respect to the metric.

5 Conclusion

In this paper, we have presented two zone-based cooperative caching protocols for MANETs. Both protocols demonstrate sensitivity with respect to the cache size. This indicates their capability for exploiting localized cooperative caching. Owing to this observation the benefits of the proposed protocols should be evident for MANETs of large scale.

References

1. R. Beraldi, R. Baldoni, "A Caching Scheme for Routing in Mobile Ad Hoc Networks and Its Application to ZRP," IEEE Transactions on Computers, Vol. 52, Issue 8, August 2003, pp. 1051-1062.
2. J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva, "A Performance Comparison of Multi-Hop Wiress Ad Hoc Network Routing Protocols," ACM MobiCom, October 1998, pp. 85-97.
3. G. Cao, L. Yin, C. R. Das. "Cooperative Cache-Based Data Access in Ad Hoc Networks," Computer, vol. 37, Issue 2, Feb 2004, pp. 32-39.
4. C. C. Chiang, H.K. Wu, W. Liu, and M. Gerla, "Routing in Clustered Multihop Mobile Wireless Networks with Fading Channel," Proc. of IEEE Singapore International Conference on Networks, 103 Singapore, April 1997, pp. 197-211.

5. C.-Y. Chow, H.V. Leong, A. Chan, "Peer-to-Peer Cooperative Caching in Mobile Environments," Proc. of the 24th ICDCS Workshops on MDC, pp. 528-533.
6. C.-Y. Chow, H.V. Leong, A. Chan, "Cache Signatures for Peer-to-Peer Cooperative Caching in Mobile Environments," Proc. of AINA 2004, pp. 96-101.
7. C.-Y. Chow, H.V. Leong, A. T. S. Chan, "Group-based Cooperative Cache Management for Mobile Clients in Mobile Environments," Proc. of ICPP 2004, pp. 83-90.
8. T. Clause, P. Jacquet, and A. Laouti, P. Minet, P. Muhlethaler, A. Qayuam, L. Viennot, "Optimized link state routing protocol," draft-ietf-manet-olsr-07.txt, IETF, http://www.ietf.org, November 2002.
9. L. Fan, P. Cao, J. Almedia, and A. Broder, "Summary cache: A scalable wide area web cache sharing protocol," ACM SIGCOMM 1998, pp. 254-265.
10. Z. J. Haas and M. R. Pearlman, "The Zone Routing Protocol (ZRP) for Ad Hoc Networks," Internet Draft, draft-ietf-manet-zone-zrp- 01.txt, 1998.
11. T. Hara, "Effective Replica Allocation in Ad Hoc Networks for Improving Data Accessibility," Proc. of IEEE INFOCOM 2001, pp. 1568-1576.
12. T. Hara, "Replica Allocation in Ad Hoc Networks with Periodic Data Update," Proc. of International Conference on Mobile Data Management, 2002, pp. 79–86.
13. T. Hara, "Replica Allocation Methods in Ad Hoc Networks with Data Update," Mobile Networks and Applications, Vol.8, No.4, August 2003, pp. 343-354.
14. M. Jiang, J. Li, and Y.C. Tay, "Cluster based routing protocol (CBRP) functional specification," Internet Drafe, draft-ietf-manet-cbrp-spec-00.txt., 1998.
15. Jie Wu and Fei Dai, "A Generic Distributed Broadcast Scheme in Ad Hoc Wireless Networks," IEEE Transactions on Computer, Vol. 53, No. 10, October. 2004, pp. 1343-1354.
16. D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," Mobile Computing, edited by Tomas Imielinski and Hank Korth, Kluwer Academic Publishers, ISBN: 0792396979, Chapter 5, 1996, pp. 153-181.
17. W. Lau, M. Kumar, and S. Venkatesh, "A Cooperative Cache Architecture in Supporting Caching Multimedia Objects in MANETs," The Fifth International Workshop on Wireless Mobile Multimedia, 2002.
18. T. Moriya and H. Aida, "Cache Data Access System in Ad Hoc Networks," Vehicular Technology Conference, Vol. 2, April 2003, pp. 1228-1232.
19. M. Papadopouli and H. Schulzrinne, "Effiects of power conservation, wireless coverage and cooperation on data dissemination among mobile devices," ACM MobiHoc, Oct. 2001, pp. 117-127.
20. C. E. Perkins and E. M. Royer, "Ad-hoc On-Demand Distance Vector Routing," Proc. of IEEE WMCSA 1999, pp. 90-100.
21. C. E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," Proc. of ACM SIGCOMM 1994, pp. 234-244.
22. A. Rousskov and D. Wessels, "Cache Digests," Computer Networks and ISDN Systems, Vol. 30, No. 22-23, 1998, pp. 2155-2168.
23. F. Sailhan and V. Issarny, "Energy-aware Web Caching for Mobile Terminals," Distributed Computing Systems Workshops, July 2002, pp. 820-825.
24. M. Tamori, S. Ishihara, T. Watanabe, and T. Mizuno, "A Replica Distribution Method with Consideration of the Positions of Mobile Hosts on Wireless Ad-hoc Networks," Distributed Computing Systems Workshops, July 2002, pp. 331-335.
25. D. Wessels and K. Claffy, "ICP and the Squid Web Cache," IEEE Journal on Selected Areas in Communication, Mar. 1998, pp. 345-357.
26. L. Yin and G. Cao, "Supporting Cooperative Caching in Ad Hoc Networks," Proc. of IEEE INFOCOM 2004, pp. 2537-2547.
27. ns Notes and Documentation. (http://www.isi.edu/nsnam/ns/)

Supporting SIP Personal Mobility for VoIP Services

Tsan-Pin Wang[1] and KauLin Chiu[2]

[1] Department of Computer and Information Science, National Taichung University, 140, Min-Shen Rd, Taichung, 403 Taiwan, R.O.C.
tpwang@ms3.ntctc.edu.tw
[2] Department of Computer Science and Information Engineering, National Chung Cheng University, 168, University Rd., Min-Hsiung Chia-Yi, Taiwan, R.O.C.
Kaulin.chiu@msa.hinet.net

Abstract. SIP is promising for VoIP signaling to support personal mobility. In this paper, we introduce and compare single registration (SR) and multiple registration (MR) for personal mobility. The SR scheme can not support personal mobility without user's assistance. In contrast, the MR scheme supports personal mobility inherently using sequential search or pure parallel search. Sequential search may suffer from long delay for call setup, while pure parallel search consumes network resource. To compromise the two schemes, we propose pipelined search for multiple registration.

1 Introduction

In early days, the key technology of VoIP was H.323 [1][2]. The H.323 standard was specified by the ITU-T Study Group 16. The advantages of H.323 include high-reliability and easy to maintain. However, H.323 still has lots of shortcomings, for example, lack of flexibility and high construction cost. Because of these shortcomings, H.323 is not deployed worldwide.

In order to solve these shortcomings the Internet Engineering Task Force (IETF) draws up a standard protocol, Session Initiation Protocol (SIP) [3]. SIP is an application-layer signaling protocol for initiation, modification, and termination of sessions with two or more participants. SIP offers a chance to realize low construction cost and high flexibility. The media stream of SIP can be video, audio or other Internet-based multimedia applications, such as white board, shared text editors, etc.

Unlike H.323, SIP is a text-based protocol similar to Hyper Text Transfer Protocol (HTTP) [4]. SIP and HTTP have a lot of similarity on processing and transmitting information. SIP continues using the request-response model, much of the HTTP syntax, header fields and semantics. Because of its simplicity and popularity, SIP has been promising in VoIP environment [5].

SIP has several key components [6], including user agents, redirect servers, proxy servers and registrars. User Agents (UAs) are endpoint devices that originate and terminate SIP requests (signaling). They can be either clients (UAC) that initiate requests or servers (UAS) that respond to the requests, or more normally a combination of both.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 703 – 714, 2005.
© IFIP International Federation for Information Processing 2005

The UAs are addressed by SIP-URLs that are similar to the email address form, for example, sip:TPwang@sip.pu.edu.tw or tel: TPwang@sip.pu.edu.tw.

Redirect Servers receive requests and push routing information for requests back in responses to the client. Registrars are special User Agent Servers that handle "REGISTER" requests. SIP users/devices use "REGISTER" requests to dynamically register their current locations. After registration, the SIP user/device can be contacted even when they move.

Typically, UAs will send a "REGISTER" message to a specific registrar server. If username in the "REGISTER" message is authorized, it can receive a final response (200 OK) and the registrar server can store user information to the location database, as shown in Fig. 1.

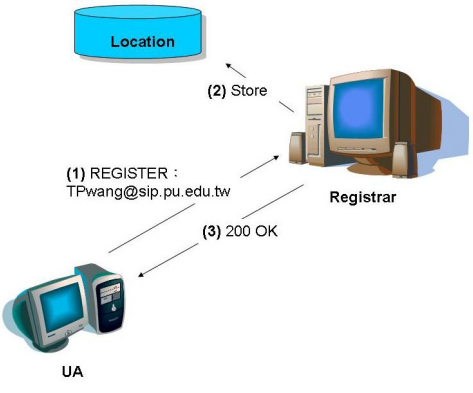


Fig. 1. Registration Scenario

Proxy Servers are elements that route requests to the user agent server and responses to the user agent client. A proxy server can operate in either a stateless proxy or a stateful proxy. A stateless proxy server just simply forwards incoming requests to another server or client, without dealing with any reliability. It forwards every request downstream to a single element determined by making a routing decision based on the request and simply forwards every response it receives upstream. In contrast, a stateful proxy maintains information (specifically, transaction state) about every received request and any responses produced by the request message that it sent.

A stateful proxy can be a *forking proxy* [4] that can route request to multiple destinations. Using forking is useful when proxy servers do not know the exact final destination. Proxy servers can either try a set of destination in pure parallel search, sequential search or other hybrid algorithms.

Practically, we can implement registrar, proxy, redirect server in the same machine, which is called "***Call Server***".

A successful SIP call invitation mush consists of two messages, an INVITE and followed by an ACK [7] [8]. The INVITE request asks the callee to join a particular conference or to establish a two party conversation. The request message's body may include some description of the session using Session Description Protocol (SDP). SDP

contains distinction address, codec, connection ports and other information. After the callee agrees and answers to join this call, the caller confirms that it has received a "200 OK" response by sending an ACK message. A success response must indicate which media type the callee wishes to receive and may indicate the media callee is going to send. Finally, the media stream will be established by using Real Time Protocol (RTP) and Real Time Control Protocol (RTCP) to transport digital audio or video.

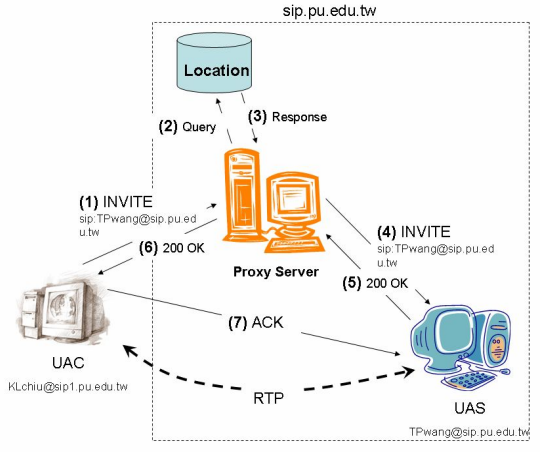


Fig. 2. Session Set-up

Consider an example of the session setup in which an INVITE message is sent from KLchiu@sip1.pu.edu.tw to TPwang@sip.pu.edu.tw, as shown in Fig. 2. Typically, all requests will be sent to a predefined local proxy server. Then the local proxy server would check the registrar's database in order to look up whether the callee is on-line or not. If the callee is found, the proxy server would forward the INVITE message to appropriate UAS. When TPwang answers the call, UAS would send a "200 OK" message to UAC via the proxy server. Finally, this call will be established using RTP protocol.

2 Single and Multiple Registration

In general, most of inter-communication platforms accept their user to register only place at the same time, for instance, MSN messenger and skype. This architecture is referred to as ***single registration (SR)***. The SR architecture does not support personal mobility inherently because the registration cannot be transparent to the user. In other word, it cannot address a single user location at different terminals using the same logical address. This way will be very inhumanity, because we cannot always ask users to sit in front of the computer or hand-carrying terminals. Meanwhile, the proxy server must accept to authorize shorter legal service time in order to alleviate the phenomenon that users have left the terminal. In RFC-3261, the value called Expire is defined to solve this problem. The default value could be 1,800 or 3,600 seconds.

In order to solve the drawback, a good solution is let all terminals of the user can register into registrar server at the same time. This method is called ***multiple registration (MR)***. Fig. 3 demonstrates an example of the contact information stored in the iptel SER's registrar [9] for multiple registration.

username	contact	cseq
0944021400	sip:0944021400@140.128.19.178:5060	130
0944021405	sip:0944021405@140.128.10.167:5060	2120
0944021400	sip:0944021400@140.128.10.99:5060	31302

Fig. 3. iptel SER's registrar for multiple registrations

Using multiple registration, the forking proxy [10] can search several destinations of the callee. Typically, there are two algorithms to search the current location of the callee: sequential search and pure parallel search. The sequential search will use First In First Service (FIFS) to determine the processing order. In worst case, this method has an important and critical shortcoming that calls setup will fail. That is, when user is near device registered recently. Therefore, a timeout mechanism will be necessary to continue searching the next possible location. In contrast, the forking proxy searches all destinations in parallel. However, the pure parallel search consumes more network resource.

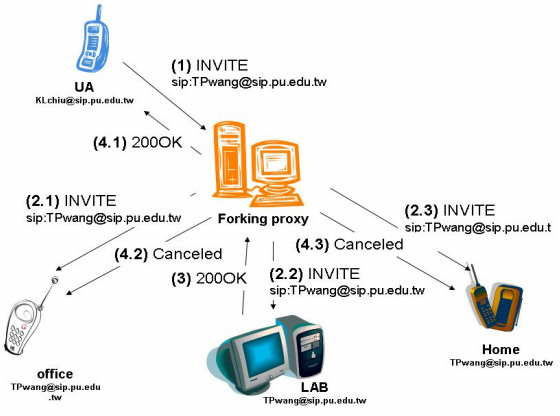


Fig. 4. Pure Parallel Search

Consider an example of the pure parallel search. Suppose that TPwang may move between three locations: LAB, office, and home as shown in Fig. 4. When KLchiu wants to make a phone call to TPwang, the forking proxy will fork three INVITE messages to all of TPwang's possible terminals at the same time if they are on-line. In this example, we assume that TPwang is at laboratory and he answers this call in the

LAB. Therefore, the session will be established from KLchiu's UA to TPwang's UA at laboratory. Finally, other INVITEs will be cancelled using CANCEL method.

3 Pipelined Search Algorithm

In this section, we propose a pipelined search scheme for multiple registration. Pipelined search is a hybrid method which combines sequential search and pure parallel search. It can compromise call setup delay and search cost at the same time. This algorithm defines a "d" parameter value which is used to delay the time of issuing the next request according to network status and user's behaviors. In most situations, d-value ranges from several hundred millions to several seconds. And, we use q-value for priority. It is possible to generate the q-values by analyzing the user's mobility behaviors. We also define "Group" for those sent together. Group members will have the same or similar q-value. The group concept is shown in Fig. 5.

We can get a priority list by calculating q-values and regulating the size of d-values to determine the way of search. When d-value is large, pipelined search is similar to the

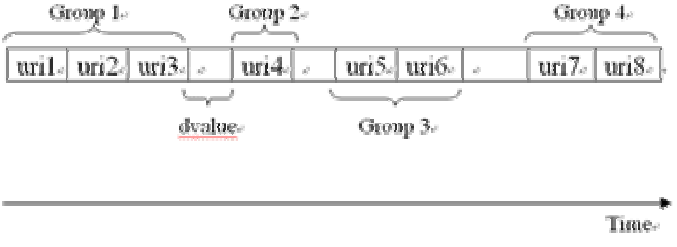


Fig. 5. The group concept

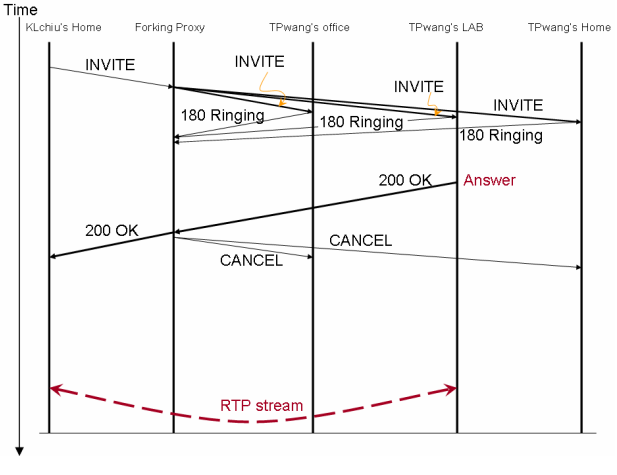


Fig. 6. Timing diagram for parallel search

sequential search. On the other hand, pipelined search is similar to pure parallel search when d-value is small.

For simplicity, in Fig. 6 and Fig. 7, we ignore some provisional response messages, and "183 Call Pregree" messages between TPwang and KLchiu's UA. When using pure parallel search algorithm, the forking proxy will receive one "INVITE", three "180 Ringing", one "200 OK", three "ACK" and forward three "INVITE", one "180 Ringing, one "200 OK", 2 "CANCEL", one "ACK". Therefore, we can derive the total number of messages sent or processed by the forking proxy as the following equation (1).

Consider the omitted provisional messages, the parallel search method will waste more resource of the network and search cost. In the piplelined search case, the phone call will be bulit as soon as possible beause of TPwang is in the LAB as shown in Fig. 7. So, forking proxy does not send the third "INVITE" to TPwang's Home and can reduce the number of sent messages.

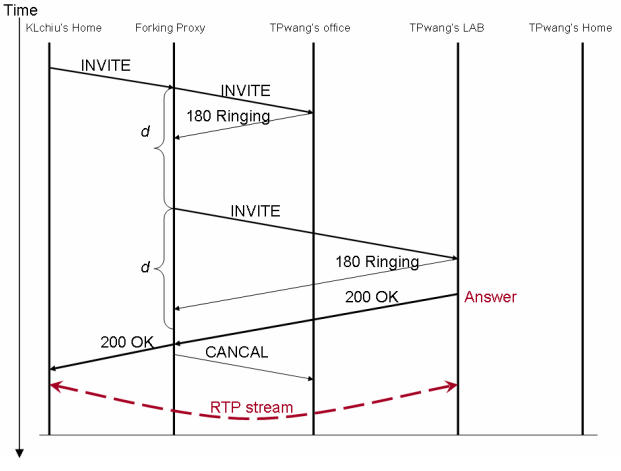


Fig. 7. Timing diagram for pipelined search

$$f(N) = 4N + 4 \tag{1}$$

where N is the number of terminals.

Moreover, if the forking proxy chooses higher q-value to send the "INVITE" message first and delays a time period of d for the subsequent "INVITE" messages. The search cost will be significantly reduced.

4 Performance Analysis and Comparison

In this section, we first compare the performance comparison of single registration and multiple registration mechanisms, and then discuss the impact of locality on multiple registration with pipeline parallel search.

4.1 Comparison of Single and Multiple Registrations

In the literature, perfornance evaulation of Mobile IP and SIP can be found in [11][12]. However, there is no research to compare single and multiple registrations for SIP personal mobility. In this subsection, we analyze and compare the performance of single registration and multiple registration.

Single registration is suitable for users with high Call-to-Mobility Ratio (CMR) and without locality behavior. It means the SR scheme is suitable for terminal mobility instead of personal mobility beause a terminal will always update its location information automatically in order to keep the lastest current position when users move from one place to another. However, supporting personal mobility in SR scheme relies on user to assist UA to send "REGISTER" message to the registrar server. If a user moves and forgets to register in the new terminal, a call to the user will fail to be delivered.

Multiple registration is suitable for users with low CMR and with regular mobility pattern with locality behavior. If the user's mobility pattern is regular, the MR scheme allows the SIP terminals to ask the forking proxy issuing a longer legal service time in the "REGISTER" message, for instance, 7200 seconds or more. In the best case, registration is necessary only in the first access. Since registration is unusual in MR scheme, the cost of registration will tend to be ignorable in the long term.

Sequential search, pure parallel search, and pipelined search can be used to search the user's current location for multiple registration. Sequential search suffers from longer delay to wait for timeout on searching the possible user location. It is unsuitable for the caller without patience to wait the long delay. On the other hand, the pure parallel search will outperform others in terms of its short delay for call setup. Because all INVITE messages will be sent at the same time, this algorithm is suitable only for the user with locality in a small number of possible locations. Otherwise, many network resources will be waste. Note that the performance of pipelined search depends on the distribution of user mobility pattern.

In the following, we derive the total cost for SR and MR schemes. In general, the total cost of a scheme is the sum of the paging/searching cost and the registration cost. The paging cost indicates the number of messages that a proxy spends for searching the user location. And, the registration cost is the messages sent to register the user location. Normally, every paging consists of eight incoming or out-coming messages that include INVITE, 100 trying, 180 ringing and 200 OK. Note that the provisional messages (100 trying and 180 ringing) are omitted in Fig.2 for simplicity. On the other hand, the registration includes two messages that are "REGISTER" and "200 OK" as shown in Fig.1.

Suppose that the call rate is λ and the mobility rate is μ for a SIP user. That is, a proxy server may perform λ times paging and receive μ times registration in a time unit. We derive two equations for the total cost of single registration and multiple registration as follows.

According to the above description, the total cost of single registration (Cost_S) is

$$Cost_S = 8 * \lambda + 2 * \mu \tag{2}$$

As we mentioned above, the cost of registration will tend to be ignorable in the long term. Therefore, the total cost of multiple registration (Cost_M) is equal to the paging cost. From Equation (1), the total cost is

$$\begin{aligned} Cost_M &= \lambda * [4N + 4] + 0 \\ &= 4N\lambda + 4\lambda \end{aligned} \tag{3}$$

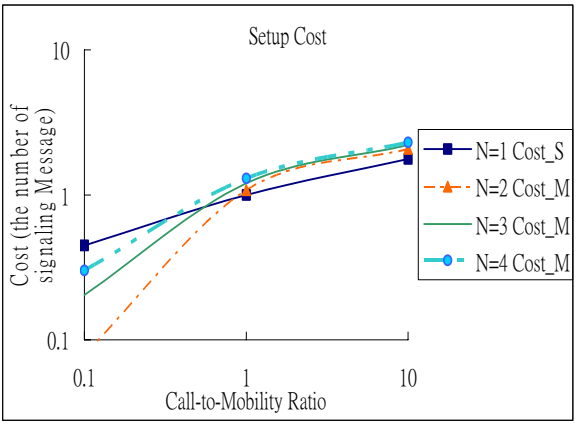


Fig. 8. The impact of CMR to call setup cost

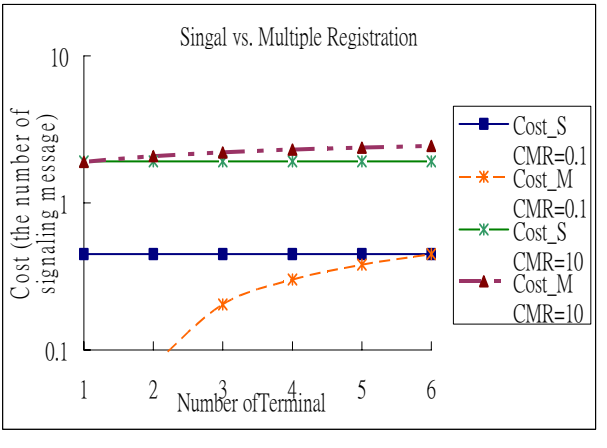


Fig. 9. The impact of n to call setup cost

In equation (3), we assume that "ACK" message will pass through the proxy server for stateful processing.

The results derived from equations (2) and (3), are shown in Fig. 8 and Fig. 9. In order to demonstrate the major difference, we perform nature logarithm treatment to the

total cost. Fig. 8 shows the impact of CMR on the total call setup cost. Note that the call setup cost increases as CMR increases. Fig. 9 shows the impact of the number of terminal to the total call setup cost. Single registration accepts only one terminal registering to the registrar at the same time. Consequently, its setup cost is always the same. It is obvious that the costs for single and multiple registrations have only little difference when the user has only two terminals. However, single registration must issues "REGISTER" request message. Therefore, its cost is higher than that of multiple registration.

4.2 Impact of Locality on Pipeline Search

After comparing MR with SR, we further discuss the impact of locality behavior on pipelined search in multiple registration based environment. Performance metrics include the mean call-setup delay and the mean number of message sent to setup a call. In addition, we consider two mobility patterns: uniform and locality distributions. Uniform distribution means the user appears uniformly in all possible locations. In contrast, locality distribution means that the user may appear in a few locations with higher probability.
Prior to deriving the results, we list the used notation as follows.

RT：Response time
d：dvalue
Pi：Probability value
N：The number of terminals
t：The time of successful setting up a call

We first derive the mean delay time (t) for setting up a call.

- **Uniform distribution**

Since the user may appear uniformly in all possible locations, the probability of the user in each location is the same. Therefore, the mean delay for call setup is

$$E(t) = \sum_{i=1}^{N} [RT + (i-1) * d] * p_i$$
$$\because p_n = p_1 = p_2 = \ldots = \frac{1}{N} \tag{4}$$
$$\therefore E(t) = RT + \frac{N-1}{2} * d$$

- **Locality distribution**

Without loss of generality, we assume that the probability (pBiB) of the user in location i is twice the probability in location i+1 (pBi+BB1B) for all possible i in locality distribution. Consequently, we can easily derive the probability pBiBB.B and the mean call setup delay as follows.

$$
\begin{aligned}
C &= 1 + 2 + 4 + 8 + \ldots + 2^{N} = 2^{n} - 1 \\
p_i &= \frac{2^{N-i}}{2^{N} - 1} \qquad 1 <= i <= N \\
E(t) &= \sum_{i=1}^{N} [RT + (i-1) * d] * p_i \\
&= RT + \frac{d}{2^{N} - 1} * \sum_{i=1}^{N-1} i * 2^{N-i-1}
\end{aligned}
\tag{5}
$$

In Fig. 10, we assume the value of RT is 3. The values in the pure parallel search are not changed. Because pure parallel search always sends to all destinations at the same time, it produces no extra delay. And, the curve with uniform distribution will grow up linearly according to the number of group. In contrast, the curve with locality distribution will lead to a fix value when N approaches 10. Note that the difference between different d values is very small.

In Fig. 11, we define the value of M as RT divided by d and N is equal to 10. The result from this figure is very close to that of pure parallel search when M is more and

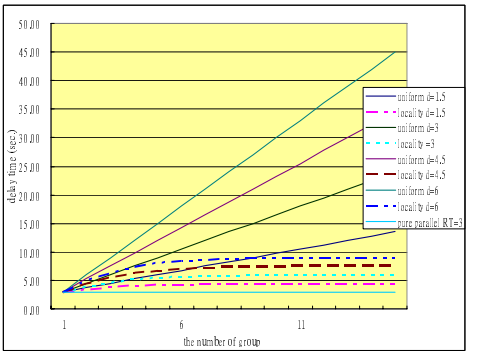


Fig. 10. The Delay Time for Invitation Call

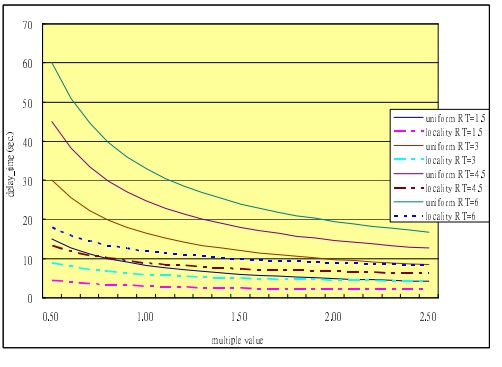


Fig. 11. The Impact of the M to Call Setup

more larger (i.e., RT is more than d) and the delay time will be reduced. In any cases, the one with locality distribution outperforms the one with uniform distribution.

In the following, we derive the mean number of messages sent for call setup. Let n denote the number of messages sent for setting up a call.

- **Uniform distribution**

$$E(n) = \sum_{i=1}^{N} p_i * [(i-1) + \left\lceil \frac{RT}{d} \right\rceil] = \frac{N+1}{2} - 1 + \left\lceil \frac{RT}{d} \right\rceil \quad (6)$$

- **Locality distribution**

$$E(n) = \sum_{i=1}^{N} p_i * [(i-1) + \left\lceil \frac{RT}{d} \right\rceil] = \frac{1}{2^N - 1} \sum i * (2^{N-i}) - 1 + \left\lceil \frac{RT}{d} \right\rceil \quad (7)$$

In Fig. 11, the locality distribution approaches to a fixed value when N is near to 10. Regardless of the paging cost (means sent messages) with uniform or locality distribution, pipeline search performs better than pure parallel search.

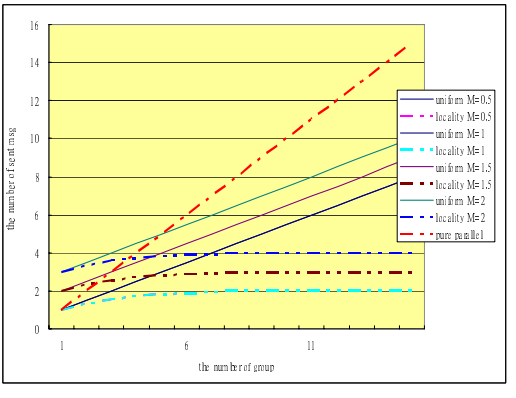


Fig. 12. Impact of the number of group

According to the above results, we observe that the one with locality distribution is very stable when N approaches to 10. It is strongly recommended that the system should provide multiple registration service for the users with locality behavior. In other hand, RT is very difficult to control in different network environment. Network manager can refer to our results (in figures 11 and 12) to adjust the d-value.

5 Conclusions

In this paper, we compare the performance of single registration and multiple registration. The single registration scheme is suitable for users with high call to mobility ratio. In contrast, multiple registration is suitable for users with low call to mobility ratio. Moreover, sequential search suffers from long setup delay, while pure parallel search consumes more resource to paging the user location. A compromise solution we propose is pipelined search. In pipeline search, if we can get a good algorithm for deriving d-value and q-value, the pipelined search would reduce wasted resource and improve system performance.

Acknowledgement

This work was supported in part by NSC under Grants NSC93-2219-E-126-001, NSC 93-2213-E-126-002, and NSC 94-2213-E-126-006, Taiwan, R.O.C.

Reference

1. ITU-T Recommendation H.323, "Packet-based multimedia communication systems", 1998.
2. B. Vlaovic, Z. Brezocnik, "Packet based telephony", EUROCON'2001, Trends in Communications, International Conference on., Vol. 1 , pp. 210 – 213, vol.14-7, July 2001
3. SIP Tutorial, at http://www.iptel.org/sip/ (last visit 2 September 2004)
4. J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002
5. K. Daniel Wong, Ashutosh Dutta, Jim Burns, Ravi Jain, Kenneth Young "A multilayered mobility management scheme for auto-configured wireless IP networks", IEEE Wireless Communications, Vol. 10, pp.62-69, Oct. 2003
6. H. Schulzrinne, E. Wedlund, "Application layer mobility using SIP",Mobile Computing and Communications Review, Volume 4, Number 3
7. E. Wedlund, H. Schulzrinne, "Mobility support using SIP", 2ndACM/IEEE International Conference on Wireless and Mobile Multimedia, Seattle, Washington, Aug. 1999
8. M. Handley, V. Jacobson "SDP: Session Description Protocol", RFC2327, April 1998
9. http://www.iptel.org/
10. Henry Sinnreich, Alan B. Johnston, "Internet Communications Using SIP", John Wiley & Sons, Inc., New York, NY, USA
11. Jin-Woo Jung, Hyun-Kook Kahng, Ranganathan Mudumbai, Doug Montgomery, "Performance Evaluation of Two Layered Mobility Management using Mobile IP and Session Initiation Protocol", at http://w3.antd.nist.gov/pubs/sip-mip-jwjung-globecom 2003.pdf
12. Peng Sun, Sam Y. Sung, Zhao Li, "Performance Evaluation and Analysis of Protocols for IP Mobility Support: A Quantitative Study", http://www.iscs.nus.edu.sg/~ssung/publications / 1011.pdf

Scalable Spatial Query Processing for Location-Aware Mobile Services

KwangJin Park[1], MoonBae Song[1], Ki-Sik Kong[1], Chong-Sun Hwang[1], Kwang-Sik Chung[2], and SoonYoung Jung[3]

[1] Dept. of Computer Science and Engineering, Korea University, 5-1, Anam-dong, Seongbuk-Ku, Seoul 136-701, Korea
{kjpark, mbsong, kskong, hwang}@disys.korea.ac.kr
[2] Dept. of Computer Science and Engineering, Korea National Open University
kchung0825@knou.ac.kr
[3] Dept. of Computer Science Education, Korea University
jsy@comedu.korea.ac.kr

Abstract. Location-aware mobile services(LAMSs) are characterized by a large number of objects and a large number of queries. Moreover, with a large candidate data set, answering LAMSs via scanning through the whole data set becomes extremely expensive. In broadcast-based services, any number of clients can monitor the broadcast channel and retrieve the data as it arrives through the broadcast channel. Thus, a wireless broadcast system capable of answering LAMSs queries is considered a promising solution because it can serve a virtually unlimited number of users within its coverage. Furthermore, if the data is properly organized to cater to the needs of the clients, such a scheme makes effective use of the low wireless bandwidth, and is ideal for achieving maximal scalability. In this paper, we address the issues of supporting spatial queries of location-aware data via wireless data broadcast. A linear data broadcast based on their location is proposed to answer spatial queries on air. Comprehensive experimentation shows that the proposed scheme is highly scalable and is more efficient in terms of both tuning time and access latency in comparison to other techniques.

1 Introduction

Data dissemination is the delivery of data from the server to a larger set of clients. This is characterized by an inherent asymmetry in the communications. Any number of clients can monitor the broadcast channel and retrieve the data as it arrives on the broadcast channel. If the data is properly organized to cater to the needs of the clients, such a scheme makes effective use of the low wireless bandwidth and is ideal for achieving maximal scalability. Two key requirements for data access in wireless environments are **the conservation of power** and **the minimization of the client waiting time**.

With a large candidates data set, answering LAMSs via scanning through the whole data set becomes extremely expensive. Thus, index structures and related

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 715–724, 2005.
© IFIP International Federation for Information Processing 2005

search algorithms have been proposed to provide efficient processing of LAMSs queries. Air indexing is one of techniques used to address this issue, and which operates by interleaving indexing information among the broadcast data items. By first accessing the broadcast index, the mobile client is able to predict the arrival time of the desired data. Thus, it can stay in the power save mode most of time, and tune into the broadcast channel only when the requested data arrives [2][3]. Air indexing techniques can be evaluated in terms of the following two factors: First, *Access Time(AT)*: The average time elapsed from the moment a client issues a query to the moment when the required data item is received by the client. Second, *Tuning Time(TT)*: The amount of time spent by a client listening to the channel. Then, the *Access Time* consists of two separate components, namely: *Probe Wait*: The average duration for getting to the next index segment. *Bcast Wait*: The average duration from the moment the index segment is encountered to the moment when the required data item is downloaded.

There are several indexing techniques which have been developed, such as the distributed indexing approach [3], the signature approach [4], and the hybrid approach [5]. In a recent paper [8], we proposed the concept of data sorting for broadcasting called BBS(Broadcast based Location Dependent Data Delivery Scheme). A preliminary simulation-based results showed that BBS is significantly reduce the *AT*. In this paper, we attempt to reduce both the *TT* and *AT* in the wireless mobile computing environment. After pointing out the limitations of the existing indexing schemes, we present various schemes that can overcome these problems. To the best of our knowledge, this is the first work in which NN query processing without on index segment is proposed and this technique provides the best access time. In this paper, we assume a geometric location model, i.e., a location is specified as a two-dimensional coordinate and the broadcasted data objects are static, such as restaurants, hospitals, and hotels. The mobile clients can identify their locations using systems such as the Global Positioning System(GPS).

The remainder of the paper is organized as follows: Section 2 provides background information. Section 3 describes the proposed algorithms. A performance evaluation is presented in section 4. Finally, section 5 concludes this paper.

2 Background

With the advent of high speed wireless networks and portable devices, data requests based on the location of mobile clients have increased in number. However, there are several challenges to be met in the development of LAMSs [1], such as the constraints associated with the mobile environment and the difficulty of taking the user's movement into account. Hence, various techniques have been proposed to overcome these difficulties.

2.1 Hilbert-Curve

In [9], a linear index structure is proposed based on the Hilbert curve, in order to enable the linear broadcasting of objects in a multi-dimensional space. The

Hilbert curve needs to allocate a sufficient number of bits to represent the index values, in order to guarantee that each of the points in the original space has a distinct value. If k is the number of bits used for a coordinate in the i-th dimension of the targeted m-dimensional space and n is the number of bits assigned to represent the coordinates, then a total of $\sum_{i=1}^{m} k$ bits need to be allocated to represent the coordinates and the expected time for the conversion is $O(n^2)$. Besides, to perform k-NN query processing using the Hilbert curve, more than two broadcast periods are needed.

2.2 Wireless Data Broadcast

The use of broadcasting methods that properly interleave index information and data on the broadcast channel can significantly improve not only the energy efficiency, but also the AT. Let N be the number of data objects and C be the download time for the required records. There are two parameters that need to be optimized in the one dimensional space of the AT and the TT [2][3]:

- Optimal Latency: The best AT is obtained when no index is broadcast along with the data. In this case, the size of the entire Bcast is minimized, and the average AT is $\frac{N}{2} + C$. However, in this case, the worst value of the average TT is obtained, since it is equal to $\frac{N}{2} + C$.
- Optimal Tune: This parameter allows the best TT to be obtained, while simultaneously increasing the AT. The server broadcasts the index at the beginning of each broadcast. In this case, the average TT is $1 + l + C$, where l is the number of levels in the multileveled index tree. In this case, the probe wait is equal to $\frac{N+index}{2}$ and the Bcast wait is equal to $\frac{N+index}{2} + C$. Since the Latency is the sum of the Probe Wait and the Bcast Wait, the average AT is equal to $(\frac{N+index}{2}) \times 2 + C = N + \text{index} + C$.

3 Proposed Algorithms

In this section, we first introduce the broadcast-based LDIS scheme(BBS) [8]. Then, we describe algorithm for selective tuning, namely Exponential Sequence Scheme(ESS).

3.1 Sequential Data Broadcast

In a recent paper [8], we have proposed the concept of data sorting for broadcasting called BBS(Broadcast-based Location Dependent Data Delivery Scheme). In the BBS method, the server periodically broadcasts the IDs and coordinates of the data objects, without an index segment, to the clients, and these broadcasted data objects are sorted sequentially, according to the location of the data objects, before being sent to the clients. The BBS provides the fastest access time, since no index is broadcasted along with the data and thus, the size of the entire broadcast cycle is minimized. A preliminary simulation-based results showed that BBS is significantly reduce the AT.

A simple sequential broadcast can be generated by linearizing the two dimensional coordinates in two different ways: i.e., horizontal broadcasting(HB) or vertical broadcasting(VB). In HB, the server broadcasts the LDD(location dependent data) in horizontal order, that is, from the leftmost coordinate to the rightmost coordinate. On the other hand, in VB, the server broadcasts the LDD in vertical order, that is, from the bottom coordinate to the top coordinate. In this paper, we assume that the server broadcasts the data objects using HB.

Notations

- O_i: broadcast data object, where $O_i \in S$
- O_{candi}: candidate for the nearest data object
- O_c: current broadcast data object (initially O_c regarded as NN), where $O_c \in S$
- q: a query point
- O_{ps}: one of the data items broadcast before O_i in the current broadcast cycle, where $O_{ps} \in S$
- O_f: the client's first tuned data item in the broadcast channel
- Data_first: the server's first broadcast data item in the current broadcast cycle.
- Flag A: if the dist(x-coordinate of O_{ps}, x-coordinate of q) is larger than dist(O_c, q), then set to 1 (initially set to 0). This flag guarantees that the client dose not miss the NN in the current broadcast channel.
- Flag B: if the dist((x-coordinate of O_c),(x-coordinate of q))< dist(O_{candi}, q)), then set to 1(initially set to 0).

There are two cases in which the clients tune to the broadcast channel.

Case 1: the client could not identify the NN in the current broadcast cycle, since it was not able to determine whether or not the desired data item was broadcasted before it tuned to the broadcast channel(see Fig 1).

Lemma 1: While Flag A=0 and $O_f \neq$ Data_first, the client could not identify the NN in the current broadcast cycle.

Proof: Let the client begin to tune at time T_i. At T_i, O_c=O_i and O_{candi} is O_i(see Fig 1). At T_{i+1}, O_c=O_{i+1} and O_{candi} is O_i, since dist(O_i, q)<dist(O_{i+1}, q). At T_{i+2}, O_c=O_{i+2} and O_{candi} is O_{i+2}, since dist(O_{i+2}, q)< dist(O_i, q) and dist(O_{i+2}, q)< dist(O_{i+1}, q). At T_{i+3}, O_c=O_{i+3} and O_{candi} is O_{i+2}, since dist(O_{i+2}, q)< dist(O_{i+3}, q). Then, the client stops tuning to the broadcast channel, since dist(O_{i+2}, q)<dist(x-coordinate of O_{i+3}, x-coordinate of q). However, the client could not guarantee O_{i+2} as the NN, since it missed the O_{i-n}, such as O_{i-1} data item, and one of them could also be O_{candi}. Thus, if Flag A=0 and Flag B=1, then the client stops tuning to the broadcast channel and switches to doze mode, until the Data_first of the next period arrives. □

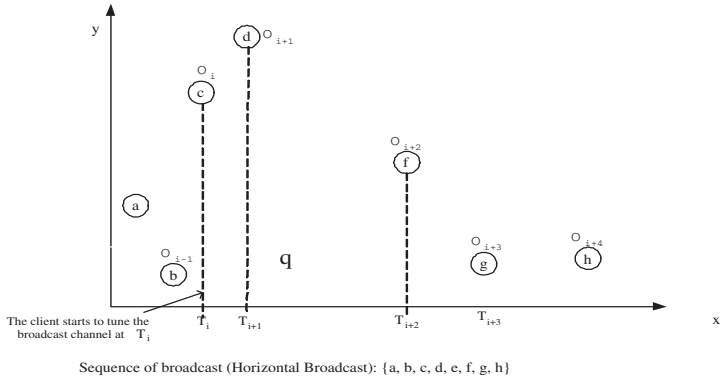


Fig. 1. The client could not identify the NN in the current broadcast cycle

Case 2: the client is able to identify the NN from the current broadcast cycle, since it is sure that the desired data item is going to appear in the current broadcast cycle(see Fig 2).

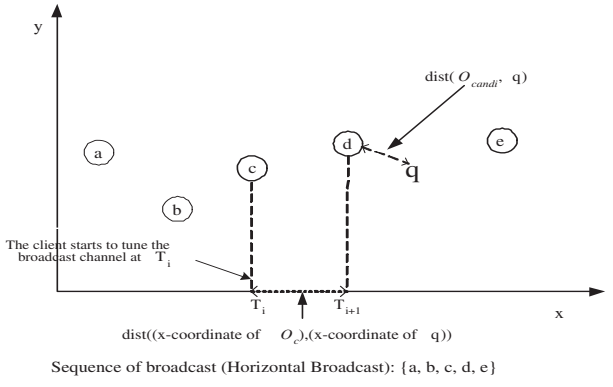


Fig. 2. The client is able to identify the NN in the current broadcast cycle

Lemma 2: If Flag A is set to 1, then the client can identify the NN from the current broadcast cycle.

Proof: Let O_c be data item 'c' in Fig 2. After the client receives data item 'd', Flag A is set to 1, which means that data item 'c' and the data items before 'c', such as 'a' and 'b', are not candidates for the NN, since the distances from the x-coordinates of 'a', 'b', and 'c' to the x-coordinate of q are longer than dist(d, q). Consequently, we can conclude that the client does not miss the NN and can find it from the current broadcast channel. □

Definition 1: If O_c is Data_first, then the client can identify the NN from the current broadcast cycle.

3.2 Energy Efficient Selective Tuning

With the BBS scheme [8], the clients can significantly reduce their access time, since this scheme eliminates the **probe wait time**(see introduction) for the clients. However, the average TT may increase, since the client has to tune to the broadcast channel until the desired data item has arrived. In the previous index schemes [3], each data item contains a pointer to the next occurrence of the index segment and, additionally, every data item contains a pointer to the next data item that has the same attribute value [3]. In our scheme, every data item contains pointers that contain the IDs, locations and arrival times of the data items that will subsequently be broadcasted. In this section, we present energy efficient scheme for the BBS environment, namely the Exponential Sequence scheme. These scheme provides the clients with the ability to perform selective tuning and help to reduce the client's tuning time.

Notations

- T_i: boundary lines of the current broadcast data object
- T: set of T_i,
- TN: nearest boundary line on the left-hand side of the q
- ON: data object of TN
- TS: safe nearest boundary line on the left-hand side of the q, where x-coordinate of $TS \leq$x-coordinate of TN

Exponential Sequence Scheme(ESS). In this section, we present a selective tuning method for use in the BBS environment. In this method, the client uses exponential pointers from each data item for the purpose of reducing energy consumption. Each data item contains the following information;

- It's ID and location information.
- Initial pointer: arrival time of the first data item to be broadcasted in the next broadcast cycle
- Forward Pointer(FP): IDs locations and arrival times of the data items that will be broadcasted at T_i. The maximal number of FP from the each data item is $\log_e N$, where e is the exponent value and N is the number of data items that will be broadcasted(e.g., if $e = 2$ and N=32, the first broadcasted data item $O1$ has the following FPs: the data items located in $T2$, $T3$, $T4$, $T5$ and $T6$(see Fig 3))

The client obtains the ID, location information and FP from the first tuned data item. Then, it switches to doze mode until the desired data item appears on the broadcast channel. The client repeatedly switches between the doze

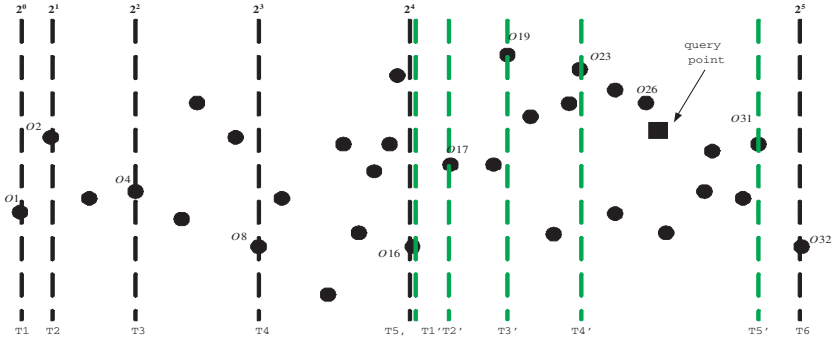


Fig. 3. An example of Exponential Sequence Scheme

and wake up modes until it finds the desired data item. Let us consider the example in Fig 3. First, the client tunes to the broadcast channel at $T1$ and obtains the following pointers ID={2,4,8,16,32} from data item $O1$. Then, it switches to doze mode until $O16$(at $T5$) has appeared on the broadcast channel, since $T5$ is the nearest boundary line on the left-hand side of the query point q up to the present time. Then, the client wakes up at $T5$, and obtains FP ID={17,19,23,31} from data item $O16$. The client again switches to doze mode until $O23$(at $T4'$) has appeared on the broadcast channel, since $T4'$ is the nearest boundary line on the left-hand side of the query point q up to the present time. Finally, the client wakes up at $T4'$ and follows the algorithm 1 and returns data item $O26$ as the NN.

The client uses the following algorithm to perform selective tuning(This algorithm is adapted to ESS).

Algorithm 1. the client algorithm used to identify the nearest object

Input: locations of the clients and the data objects;
Output: NN;
Procedure:

```
1: do
2:    read O_c
3:        for(T)
4:            if(T_i=T_N)
5:                then T_i=T_{i-1}
6:                    do compare(dist(ON, q) and dist(x-coordinate of T_i, x-coordinate of q)
7:                        if(dist(ON, q)<dist(x-coordinate of q)
8:                            then TS=T_{i-1} and the client goes into doze mode
                              until the data object of TS appears on the channel
9:                        else T_i=T_{i-1}
10:                   while find out TS
11:           else T_i=T_{i+1}
12: while(x-coordinate of T_1≤x-coordinate of ON) // T_1= TN
13: if(x-coordinate of T_2≤x-coordinate of ON)
14:  then
15:       do compare dist(O_c, q) and dist(O_p, q)
16:           if(dist(O_c, q) < dist(O_p, q))
```

```
17:              then O_c = O_candi
18:          else O_p = O_candi
19:      while dist((x-coordinate of O_c),(x-coordinate of q))< dist(O_candi, q))
20:  O_candi = NN
21:  return NN
```

4 Performance Evaluations

In this section, we evaluate the performance with various kinds of parameter settings.

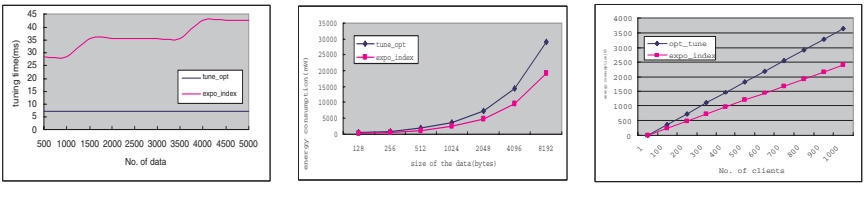
4.1 Energy Consumption Model

Let $\mathcal{E}$ denotes the amount of energy consumption and the "ec" denotes the energy coefficient which means the active-to-doze ratio, $\frac{\mathcal{E}_{ACTIVE}}{\mathcal{E}_{DOZE}}$.

In this paper, the average energy consumption can be measured by the amount of unit energy in a given time. In order to choose reasonable coefficients, we should have some reference values. Then, the energy coefficients($\widehat{ec}$) can estimated. In our experiment, parameters ec is fixed with 48.61.

4.2 Experimental Evaluation

This section presents the numerical results with the various environmental variables. We assume that the broadcast data objects are static, such as restaurants, hospitals and hotels. In this paper, two datasets are used in the experiments. The first data set, $\mathcal{D}1$, contains data objects randomly distributed in a square Euclidian space, while the second data set, $\mathcal{D}2$, contains the data objects of hospitals in the Southern California area, and is extracted from the data set available at [7].

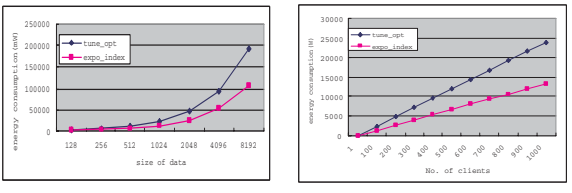


(a) (b) (c)

Fig. 4. Tuning time and energy consumption

4.3 Tuning Time and Energy Consumption

Fig 4(a) shows the tuning time as the number of data items increases. As shown in this figure, tune_opt outperforms the other schemes, since the average tuning time of tune_opt is $1 + l + C$ [3]. Fig 4(b) shows the energy consumption as the size of the data items is increased from 128 bytes to 8192 bytes in $\mathcal{D}1$. As shown in this figure, in this case, the proposed schemes outperform tune_opt, since tune_opt minimizes the tuning time but not the access time, while the proposed schemes reduce the tuning time and AT at the same time. In other words, when we estimate the energy consumption, it is necessary to consider not only the active time, but also the doze time, even if the doze time is considerably smaller than the active time(i.e., we assume a ratio of $\frac{1}{48.61}$). Fig 4(c) shows the energy consumptions as the number of clients is increased in $\mathcal{D}1$. Figs 5(a) and 5(b) show the energy consumptions as the size of the data items and the number of clients are increased in $\mathcal{D}2$, respectively.

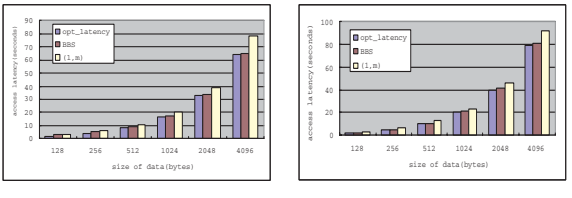


(a) (b)

Fig. 5. Energy consumption in $\mathcal{D}2$

4.4 Access Time

In this section, we evaluate the AT. Fig. 6(a) shows the AT as the size of the data $\mathcal{D}1$. Finally, Fig. 6(b) shows the AT as the items increase in size of the data items increase in $\mathcal{D}2$. Since the clients do not need to wait and tune into the broadcast channel to receive an index segment, if they have already identified the desired objects, the BBS scheme shows increasingly lower latency compared to the other scheme.



(a) (b)

Fig. 6. Access time in $\mathcal{D}1$ and $\mathcal{D}2$

5 Conclusion

We explored the different broadcasting and tuning methods which can be used for NN query processing. For the purpose of broadcasting in LAMSs, we present the BBS method, and for the purpose of selective tuning with the BBS method, we present the ESS method. The BBS method attempts to reduce the AT, and the ESS method attempts to conserve battery power. The experimental results show that the proposed ESS scheme significantly reduces not only the AT, but also the energy consumption, since the client does not always have to wait for the index segment.

References

1. D. L. Lee, J. Xu, and B. Zheng, "Data Management in Location-Dependent Information Services," *IEEE Pervasive Computing*, 1(3), 2002.
2. T. Imielinski, S. Viswanathan, and B.R.Badrinath, "Energy efficient indexing on air," *Proc. of SIGMOD*, 1994.
3. T. Imielinski, S. Viswanathan, and B.R.Badrinath, "Data on Air: Organization and Access," *IEEE Trans. Knowledge and Data Eng*, 9(3), 1997.
4. W. C. Lee and D. L. Lee, "Using signature techniques for information filtering in wireless and mobile environments," *Distributed and Parallel Databases*, 4(3), 1996.
5. Q. L. Hu, W.-C. Lee, and D. L. Lee, "A hybrid index technique for power efficient data broadcast," *Distributed and Parallel Databases*, 9(2), 2001.
6. B. Zheng, J. Xu, and D. L. Lee, "Cache Invalidation and Replacement Strategies for Location-Dependent Data in Mobile Environments," *IEEE Trans. Comp.*, 51(10), 2002.
7. Spatial Datasets, `http://dias.cti.gr/ ytheod/research/datasets/spatial.html`.
8. K. Park, M. Song, and C.-S. Hwang, "An Efficient Data Dissemination Schemes for Location Dependent Information Services," In *Proc. of ICDCIT*, 2004. Lecture Notes Computer Science.
9. B. Zheng, W.-C. Lee, D. L. Lee, "Search Continuous Nearest Neighbor on Air," In *Proc. The First International Conference on Mobile and Ubiquitous Systems: Networking and Services*, Boston, Massachusetts, August 2004.
10. M.-S. Chen, P. S. Yu, and K. L. Wu, "Optimizing index allocation for sequential data broadcasting in wireless mobile computing," *IEEE Trans. on Knowledge and Data Engineering (TKDE)*, 2003.
11. N. Roussopoulos, S. Kelley, and F. Vincent. "Nearest neighbor queries," *Proc. of SIGMOD*, 1995.

Exploiting Mobility as Context for Energy-Efficient Location-Aware Computing

MoonBae Song, KwangJin Park, and Ki-Sik Kong

Dept. of Computer Science and Engineering, Korea University,
5-1, Anam-dong, Seongbuk-Ku, Seoul 136-701, Korea
{mbsong, kjpark, kskong}@disys.korea.ac.kr

Abstract. The "mobility" has two important aspects: i) how to support mobility and ii) how to exploit mobility. This paper considers the latter, while many existing works only consider the former. This work is trying to prove that system performance will be greatly improved by understanding a user's movement. In this paper, we propose a novel location update protocol, called SLUP which minimizes the energy consumption of a mobile client by exploiting a syntactic information of the user's movement. This concept is called *mobility-awareness*. Moreover, there are three variations of the proposed protocol in terms of choosing optimal state: SLUP/BS, SLUP/UITR, and SLUP/IUT$\langle T_{iut} \rangle$. Experimental results show that the proposed protocol outperforms the well-known protocols such as dead-reckoning and distance-based protocol, that the SLUP/IUT$\langle T_{iut} \rangle$ approach can achieve difference performance tradeoffs between energy efficiency and location imprecision by fine-tuning its algorithmic parameter T_{iut}.

1 Introduction

Definition in *Wikipedia* says that "context includes the circumstances and conditions which *surround* an event". Analyzing contextual information of human behavior from low-level sensor data is the most critical issue in context-aware computing.

Recently, the evolution of mobile computing, location sensing, and wireless networking has created a new class of computing: *location-aware computing* (LAC) [3]. A location-aware system is one that knows where each user is located and can use the location-specific information anywhere and anytime. LAC can be viewed as a kind of context-aware computing based on a context of the very "location". While location is only one small part of a user's state, location is a easily-detected and commercially-useful context. In LAC environments, the mobility of a mobile client (MC) is emerging in many forms and applications such as database, network and so on. MCs can dynamically change their locations over time. A central problem in LAC is the determination of physical location and the transmission of the location information to a server.

Another major issue in LAC is energy efficiency because of a large population of portable, battery-powered devices. Although the improvement of other

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 725–734, 2005.
© IFIP International Federation for Information Processing 2005

hardware is governed by *Moore's Law*, the lifetime of a battery is expected to increase only 20% over the next 10 years [4]. As a result, attention to power consumption must span many levels of hardware and software to be fully effective. Especially, the location update protocol is ever-running process for MCs. Therefore, it is very important to achieve energy efficiency. Basically, the energy efficiency in location update protocol can be done by (1) *reducing the number of location update message as possible* and (2) *switching to doze mode as possible*. These two approaches need also awareness of user movement.

However, this movement information used to be hidden by the process of abstraction. As stated in [10], the traditional way of treating mobility is "abstracting-it-away" method. In this method, the mobility information is abstracted in the network layer and hidden from applications in the paradigm of layered abstractions. In [10], M. Grossglauser claims that "if we break the traditional networking abstractions, mobility becomes an opportunity".

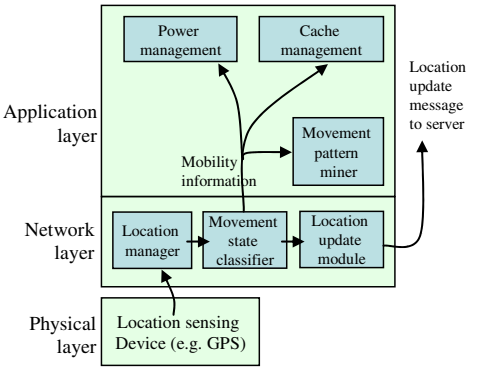


Fig. 1. The Proposed Framework for Mobile Clients in Location-aware Computing

This work is trying to prove that system performance will greatly improved by understanding the user's movement. From the viewpoint of information processing, this issue can help to solve many important issues in LAC: energy efficiency, query processing, caching, indexing, and so forth. Therefore, we can imagine a mobility-aware framework for mobile clients in LAC (Fig. 1). In this figure, the mobility information is not abstracted in the network layer, but forwarded to the application layer in a processed form which is directly exploitable in upper layer applications.

2 Backgrounds

The "location" has two important aspects: i) data that continuously changing and ii) context. In traditional DBMS's, data is assumed to be constant unless it is explicitly modified by an update operation. The former case is location of moving objects that is changing continuously. Thus, in order to represent moving objects (e.g. cars) in a database, the car's location has to be continuously updated. This is unacceptable since either the location is updated very frequently, or the answer to a query is outdated. Furthermore, it is possible that due to disconnections an object cannot continuously update its location. This needs a different approach, called moving objects databases (MOD). In [8], the authors propose a data model called Moving Objects Spatio-Temporal. Its main contribution is the concept of *dynamic attributes*, i.e. attributes that change continuously as a function of

time, without being explicitly updated. Research in MOD focuses on ways to store, index, and retrieve objects and their continuously changing locations in an efficient manner. In the latter case, there are two examples such as location-dependent mobile query (LDMQ) and *location-awareness*. The LDMQ is a query which has different results according to a user's current location. An typical example of the query is "Find a nearby restaurant". Similar to the LDMQ, *location-awareness* means that mobile devices perform different operations based on a user's location. If a user moves into a movie theater, then the mobile device of the user automatically switch into a vibration mode.

The "mobility" also has two important aspects: i) how to support mobility and ii) how to exploit mobility. Supporting the mobility is one of the key problems in mobile computing in order to enable mobile users to communicate anytime and anywhere without the restriction of geographical locations. The key technologies for communicating with mobile users include WLAN, PCS, and Mobile IP. Depending on where it is working on, the "mobility" could have various forms: component mobility, terminal mobility, user mobility, session mobility, role mobility, and application mobility. As a relatively new field of research, the latter is to empirically investigate various benefits by exploiting a user's movement behavior or pattern.

3 Approach

3.1 Basic Design Principles

Basically, the energy efficiency in mobile computing environments can be done by the following two principles.

- *Reducing the Number of Location Update Messages*: Transmitting a message consumes more power than receiving. Moreover, this power grows as the fourth power of the distance between the client and the server [4]. In the viewpoint of location update policy, the uplink message is issued when the difference between actual location and database location exceeds a predefined threshold δ[1]. Reducing the imprecision is, therefore, reducing the number of update messages as possible.
- *Switching to doze mode*: The simplest movement is "don't move"; we called the 'pause' mode. To the best of our knowledge, few studies on the location update policy for stationary state found in the literature. For this reason, every MC should operate even in stationary state. Under the pause mode, the moving object is perhaps located at his/her home, office, or meeting room for a long time. By exploiting the stationary state, location tracking for a stationary host can be done in doze mode. The *doze mode* has extremely low power consumption. The network interface card (NIC) can neither transmit nor receive until it is woken up. Then, how to decide whether an MC switch-off itself or not? For this purpose, an MC should be aware of its movement.

[1] The database location is the location information stored in the database. The difference and its threshold are called "imprecision" and "uncertainty" respectively.

3.2 Initial Experiment and Mobility-Awareness

Firstly, we introduce a new criterion to compare the efficiency of update protocols using a simple formula by measuring the update cost and the imprecision cost for a certain amount of time. This criterion is called $\mathcal{UITR}$ (*update-and-imprecision to time ratio*). $C_{\mathcal{UITR}}(\Delta, w_u, w_\epsilon) = \frac{\sum_{k=1}^{\Delta}(w_u u_k + w_\epsilon \frac{\epsilon_k}{\delta})}{\Delta}$. The $C_{\mathcal{UITR}}$ model can be interpreted as an integration of location update cost and imprecision cost, and an indirect approach to measure energy consumption by setting $w_u \gg w_\epsilon$. Based on this cost model, we can identify the behavioral difference of existing location update policies. Fig. 2 shows the comparison of distance-based approach and dead-reckoning in terms of $C_{\mathcal{UITR}}$ model. In this comparison, the movement of whole objects is random-walk or linear mobility patterns by turns with a fixed time interval 200. For random-walk, the distance-based approach outperforms the dead-reckoning in which the frequent update of motion vector may cause the increased location imprecision and the resulting increased update cost as well. For linear movement, in contrast, the dead-reckoning approach performs very well. But, the performance of the distance-based approach extremely degraded, since average movement distance per unit time highly increased.

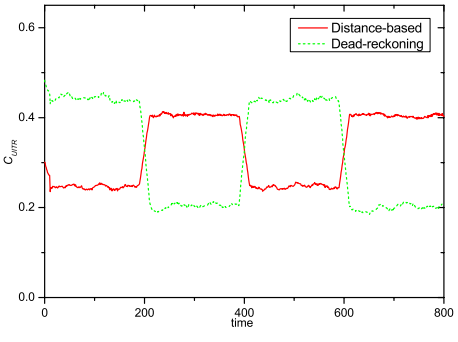


Fig. 2. An example variations of $C_{\mathcal{UITR}}(5, 1, 0.1)$ for two distance-triggered approaches ($\delta = 1.0$)

This initial experiment shows that the *mobility-awareness* is the key to reducing the number of location update messages. Exhibiting complementarity with respect to mobility patterns, different – *mutually complement* – update policies can be applied to the aforementioned states: pause (P), linear (L), and random (R). As shown in Fig. 2, a trajectory can be interpreted differently depending on various location update protocols. These interpretations are highly depends on their assumed mobility patterns. The tightness between the performance of location update protocols and their assumed mobility patterns is depicted in the figure. The efficient operations of location update protocols can be achieved by understanding user's movements. Eventually, an MC should be aware of its own mobility. This is what we called *mobility-awareness*.

Based on the initial experiment, therefore, we attempt to classify the movement pattern into a set of small movement components in the viewpoint of location update policy. This serves as a foundation of mobility-awareness. Consequently, the movement pattern is classified into *pause*, *linear movement*, and *random movement*. This classification is ambiguous/relative and may cause a different result from the different location uncertainty. In this paper, we only consider the performance viewpoint of location update policy. In other words,

the syntactic information[2] of current movement state is indirectly recognized by comparing the performance of each location update policy. The concept of *mobility-awareness* is, therefore, a process of inferring the syntactic characteristics of a movement pattern relying on the performance of location update policy.

4 The Proposed Protocol

4.1 Protocol Description

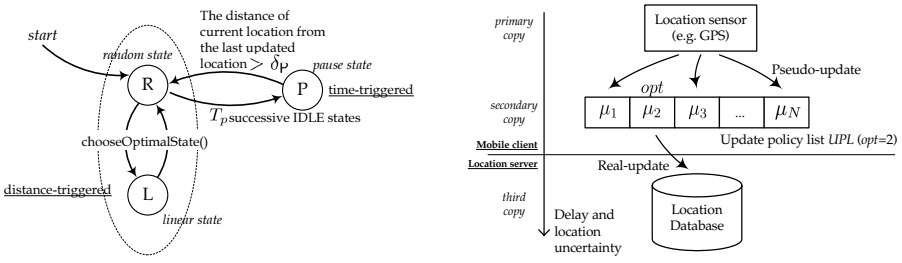
After power-up, each moving object makes transitions through various states depending on its mobility patterns, and updates its location information in location databases using the current update policy (Fig. 3(a)). Because of having more than one update policy, the proposed protocol is called state-based location update protocol (SLUP). The proposed SLUP protocol is represented by a finite set of update policy called *update policy list* $\mathcal{UPL} = \{\mu_1, \mu_2, \ldots, \mu_N\}$ and the optimal update policy index *opt*. Each update policy μ a 6-tuple $(\hat{l}, f, C, UWin, IWin, \delta)$ consisting of the estimated location $\hat{l}$ by f, a location estimation function f, the cost function C, the update window $UWin$, the imprecision window $IWin$, and a predefined location uncertainty δ. The object should examine the $\mathcal{UITR}$ value of all update policies in order to determine an optimal policy over the whole set of update policies. For this purpose, each moving object performs not only the current *opt* update policy but also the others by the pseudo-update procedure which is a self-initiated update procedure within MC's local memory (see Fig. 3(b)). Then the optimal update policy with the minimum $\mathcal{UITR}$ can be decided without any difficulty. The additional cost, a few memory and a small number of operations, is acceptable owing to its reflective effectiveness in the number of update messages and the development of hardware technology.

The behavior of a moving object is modeled as a state-transition diagram. Exploiting temporal locality of mobility patterns, the update policy phase is decomposed into small fraction of *update state* such as P, L, and R. Each update state consists of an update policy that is how the location information of an object is reflected in the location databases, the state-transition function determining the next states of the object, and information related to the state.

From the starting state R, location update is performed with state transition through P, L, and R. If the optimal state is P, then the SLUP protocol will be in time-triggered manner in period $\hat{T}$ and be in the doze mode in the remainder of time. Otherwise, it will performs a distance-triggered manner with distance threshold δ. If the *opt* state is stale[3], then call *chooseOptimalState*() to choose a

[2] In our work, the syntactic information is one of three movement states $\{\mathsf{P}, \mathsf{L}, \mathsf{R}\}$. Similar to our work, Patterson et al. [6] presented a method to learning how a person uses different kinds of transportation in the community. In this work, the mode of transportation is represented as one of three different values $\{\mathsf{BUS}, \mathsf{FOOT}, \mathsf{CAR}\}$.

[3] A state is called *stale* (or inconsistent) if the location information of a state has more location imprecision than a predefined uncertainty δ.



(a) The state transition diagram (b) The flow of location update

Fig. 3. The Proposed Protocol

new (predicted) optimal state and transmit location update message. Otherwise, check the current *opt* is R and still remains for a certain amount of time T_p, then state station to the state P is performed.

Under the pause mode, the moving object is perhaps located at his/her home, office, or meeting room for a long time. It is quite reasonable to consider a location update policy for stationary state as well as moving state. This insight has been tried in the location management of PCS, called *Stop-or-Move* mobility model [9]. We can assume that an object is in a stationary state if the moving object remains still its location for a certain amount of time T_p. The proposed approach performs location updates and makes transitions through different energy states depending on user's movement. These energy states are DOZE, IDLE, and TRANSMIT. In Fig. 4, behavioral differences of each device in terms of various energy states is presented.

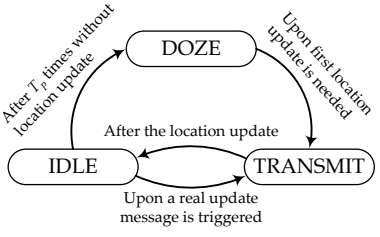


Fig. 4. Energy state transition

4.2 Variations: Choosing the Optimal State

The performance of proposed scheme is greatly dependent on determining the optimal state. When an location update message is sent, a MC should determine whether the location update needs a state transition or not. In order to determine the next *opt* state, *chooseOptimalState*() is called. And there are several variations on the subject.

In Fig. 5, we show three variations. Firstly, one of most simplest approach is "blind switching" (SLUP/BS) in which state transition is required to be done for every location update. With the location update message m_1 in Fig. 5, state transition from L to R is performed concurrently and unconditionally. This is simple, but it is more likely to make a false state transition. The second variation is called SLUP/UITR in which a state with lower $\mathcal{UITR}$ cost among states L and R is selected. In Fig. 5, assuming that the state R is more optimal one than the

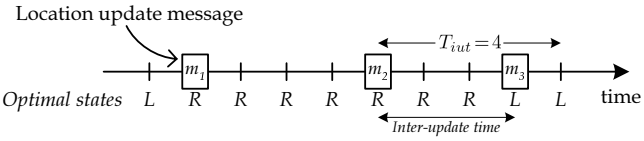


Fig. 5. State Transition Problem: To make or not to make the state transition?

state L, a state transition is not required to issuing the message m_2. The third approach considers not a single location update, but a time interval between two consecutive location updates. Let "inter-update time" be a time interval between two consecutive location updates. A state transition will be delayed until the time interval is smaller than a predefined threshold T_{iut}. In Fig. 5, state transition is made with the message m_3, because the time interval between m_2 and m_3 is smaller than $T_{iut}(4)$. This approach is called SLUP/IUT$\langle T_{iut} \rangle$. Obviously, its performance highly depends on T_{iut}. Consequently, the variants of SLUP protocol are choices on the diversity in the degree of laziness for state transition. Such a "laziness" to the location update is very helpful to the energy efficiency.

5 Performance Evaluation

5.1 Simulation Model and Workload Generation

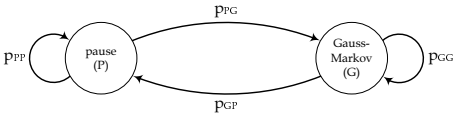
We evaluate the performance of three traditional approach, and four proposed approaches with variation. The traditional approaches include distance-based and dead-reckoning. The proposed approaches include SLUP/BS, SLUP/UITR, and SLUP/IUT$\langle T_{iut} \rangle$ with different T_{iut} (2,5).

Since the real datasets in spatio-temporal database are very hard to achieve, the method of synthesizing data has widely been used in various area. In this experiment, we adopt the Gauss-Markov mobility model [11]. Specifically, the speed and direction at the n-th instance is calculated based upon the value of speed and direction at the $(n-1)$-th instance and a random variable using the following equations: $s_n = \alpha s_{n-1} + (1-\alpha)\bar{s} + \sqrt{1-\alpha^2} s_{x_{n-1}}$ and $d_n = \alpha d_{n-1} + (1-\alpha)\bar{d} + \sqrt{1-\alpha^2} d_{x_{n-1}}$, where s_n and d_n are new speed and direction of mobile node at time interval n; $0 \leq \alpha \leq 1$, is the tuning parameter used to vary the randomness; and $s_{x_{n-1}}$ and $d_{x_{n-1}}$ are random variables from a Gaussian distribution. By setting $\alpha = 0$ we get a very random motion, while $\alpha = 1$ gives a completely linear motion. Intermediate levels of randomness are obtained by varying the value of α between 0 and 1. In our experiment, mean speed $\bar{s}$ is 1, mean direction $\bar{d}$ is randomly selected from $[0, 2\pi]$, and α is 0.9.

In order to generate more realistic movements, we also consider stationary state, called pause (P). In conjunction with Gauss-Markov model (denoted as G), these are constructed as a form of Markov chain (see Fig. 6(b)). Then, for the

Parameters	Values Used
δ (distance threshold)	2.0
τ (temporal locality)	$0.0 \sim 1.0$
ρ (dominancy)	$0 \leq \rho \leq \infty$
$(T_p, \hat{T})$	(5,5)

(a) Simulation Parameters



(b) State Diagram for Workload Generation

Fig. 6. Simulation Parameters and State Diagram for Workload Generation

purpose of generating synthetic movements, we construct two types of transition matrices: $\mathbf{T}(\tau)$ and $\mathbf{P}(\rho)$. These are shown in below:

$$\mathbf{T}(\tau) = \begin{array}{c} \\ \mathsf{P} \\ \mathsf{G} \end{array} \begin{array}{c} \begin{array}{cc} \mathsf{P} & \mathsf{G} \end{array} \\ \begin{pmatrix} \tau & 1-\tau \\ 1-\tau & \tau \end{pmatrix} \end{array}, \quad \mathbf{P}(\rho) = \begin{array}{c} \\ \mathsf{P} \\ \mathsf{G} \end{array} \begin{array}{c} \begin{array}{cc} \mathsf{P} & \mathsf{G} \end{array} \\ \begin{pmatrix} \frac{\rho}{\rho+1} & \frac{1}{\rho+1} \\ \frac{\rho}{\rho+1} & \frac{1}{\rho+1} \end{pmatrix} \end{array} \tag{1}$$

The matrix $\mathbf{T}(\tau)$ is for observing the variation of temporal locality. The temporal locality is denoted as τ which is defined as $\sqrt{p_{\mathsf{PP}} p_{\mathsf{GG}}}$. And the matrix $\mathbf{P}(\rho)$ represents how much the state P has an influence on the whole movement pattern. This measurement is called stationarity ρ, which is defined as $\frac{\sum p_{i\mathsf{P}}}{\sum p_{i\mathsf{G}}}$. Fig. 6(a) summarizes setting for the parameters.

5.2 Energy Consumption Model

In this paper, there are three energy states such as *doze* mode, *idle* mode, and *transmission* mode. We now describe the ratio of energy consumption for these states. $\mathcal{E}_s$ describes the amount of energy consumption in an energy state s per unit time. $\mathcal{E}_{\mathsf{DOZE}} : \mathcal{E}_{\mathsf{IDLE}} : \mathcal{E}_{\mathsf{TRANSMIT}} = 1 : ic : tc$. In this paper, the amount of energy consumed in doze mode for unit time is denoted as *unit energy* which is 33.16 mW in our experiment. In many processors, the doze mode has extremely low power consumption. In the Hobbit chip from AT&T, for example, the ratio of power consumption in the active mode to the doze mode is 5,000 [4]. In brief, the "ic" stands for idle coefficient which means the idle-to-doze ratio, $\frac{\mathcal{E}_{\mathsf{IDLE}}}{\mathcal{E}_{\mathsf{DOZE}}}$. Similarly, the "$tc$" stands for transmission coefficient which means transmission-to-doze ratio, $\frac{\mathcal{E}_{\mathsf{TRANSMIT}}}{\mathcal{E}_{\mathsf{DOZE}}}$.

In this paper, the average energy consumption can be measured by the amount of unit energy in a given time. Undisputedly, this experiment depends sensitively on the pair of (ic, tc). In order to choose reasonable coefficients, we should have some reference values. Table 1 shows the parameters of energy consumption for our experiment [5]. Then, the energy coefficients can estimated. $\widehat{ic} = \frac{(400+750+462)\ mW}{(0.16+0+33)\ mW} = 48.61$, and $\widehat{tc} = \frac{(400+1{,}500+462)\ mW}{(0.16+0+33)\ mW} = 71.23$. Thus, the parameters ic and tc is fixed with $(48.61, 71.23)$.

Table 1. The components of Example Mobile Client we examined (in mW) [5]

	Model	doze	normal/active	receive	transmit
CPU	StrongARM SA-1100	0.16	400 (2,500)		
NIC	RangeLAN2 7401/02	25		750 (30)	1,500 (60)
GPS	μ-blox GPS-MS1E	33	462 (14)		

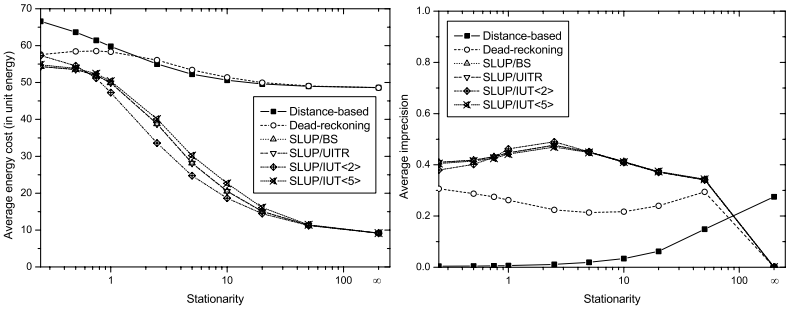


Fig. 7. Effect of stationarity ρ: energy cost and imprecision cost

5.3 Effect of Stationarity ρ and Temporal Locality τ

In this experiment, we vary stationarity ρ and temporal locality τ for observing the variety of moving behavior of mobile clients. In Fig. 7, we consider user's stationarity ρ from 0 to ∞ and location uncertainty δ is 2.0. When ρ is 0, all moving objects always moving. If ρ is ∞, all MCs stay at their initial position. As shown in the figure, the energy efficiency has increased as the parameter ρ increased. In the case of $\rho = \infty$, all objects is always on the move and the performance of the proposed approach is approximated to $\frac{ic+\hat{T}-1}{\hat{T}}$. Moreover, the performance of distance-triggered approaches is approximated to ic. Such efficiency inherently leads some additional cost of increased location imprecision. However this overhead is quite smaller than predefined uncertainty δ.

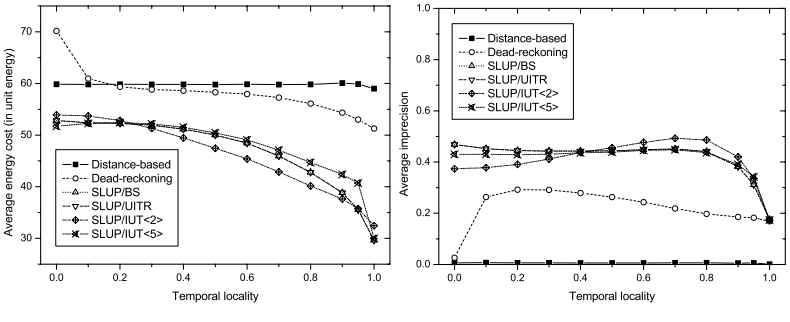


Fig. 8. Effect of locality τ: energy cost and imprecision cost

In Fig. 8, we examine the impact on the temporal locality τ of user movement. Due to fixed $\rho = 1$, the transition matrix with higher τ is likely to have more stationary or linear movements than the opposite one. The dead-reckoning approach, therefore, will be outperform distance-based approach for a higher τ.

6 Conclusions

Context-awareness has motivated various research topics. As a branch of the concept, this work is trying to prove that system performance will greatly improved by understanding the user's movement which is called *mobility-awareness* in the paper. On the basis of our experiment, we argue that the proposed concept, *mobility-awareness*, is the key to reducing the energy consumption of mobile clients. This paper also presents a novel location update protocol, called SLUP which minimizes the energy consumption of a mobile client by exploiting a syntactic information of the user's movement. Moreover, there are three variations of proposed protocol in terms of choosing optimal state: SLUP/BS, SLUP/UITR, and SLUP/IUT$\langle T_{iut} \rangle$.

References

1. A. Bhattacharya and S. K. Das. "LeZi-update: An information-theoretic framework for personal mobility tracking in PCS networks," *Wireless Networks*, 8, 2002.
2. A. Bar-Noy, I. Kessler, and M. Sidi. "Mobile Users: To Update or Not To Update?," *Wireless Networks*, 1(2), 1995.
3. Computer Science and Telecommunications Board (CSTB), National Research Council. *IT Roadmap to a Geospatial Future*, The National Academies Press, 2003.
4. T. Imielinski, S. Viswanathan, and B.R. Badrinath. "Data on Air: Organization and Access", *IEEE Trans. on Knowledge and Data Engineering*, 9(3), 1997.
5. O. Kasten, Energy consumption. Available at `http://www.inf.ethz.ch/~kasten/research/bathtub/energy_consumption.html`.
6. Donald J. Patterson, Lin Liao, Dieter Fox, and Henry A. Kautz. "Inferring High-Level Behavior from Low-Level Sensors", *Proc. of Ubicomp* 2003.
7. MoonBae Song, JeHyok Ryu, SangKeun Lee, and Chong-Sun Hwang. "Considering Mobility Patterns in Moving Objects Database," *Proc. of ICPP*, October 2003.
8. O. Wolfson, A. P. Sistla, S. Chamberlain, and Y. Yesha. "Updating and Querying Databases that Track Mobile Units," *Distributed and Parallel Databases*, 7(3), 1999.
9. Y.-C. Tseng, et al. "A Stop-or-Move mobility model for PCS networks and its location-tracking strategies," *Computer Communications*, 26:1288–1301, 2003.
10. Matthias Grossglauser, "Mobility in Ad Hoc Networks: Friend or Foe?", *Self-organized Wireless Communication Systems*, I&C Research Day 2004.
11. B. Liang and Z.J. Haas, "Predictive distance-based mobility management for PCS networks", In *Proc. of INFOCOM*, 1999.

Mobile User Data Mining: Mining Relationship Patterns

John Goh and David Taniar

School of Business Systems,
Monash University, Clayton Vic 3800 Australia
{Jen.Goh, David.Taniar}@infotech.monash.edu.au

Abstract. Mobile user data mining focuses on finding useful and interesting knowledge out from raw data collected from mobile users. Frequency pattern and location dependent mobile user data mining are among the algorithm used in this field. Parallel pattern, our previous proposed method, extracts how a group of mobile users makes similar decisions, such as by moving towards the similar direction, or by viewing similar contents at the same time. Parallel pattern is triggered group behaviour of mobile users. This paper reports our refinement work on parallel pattern which incorporated *refinement of the relationships among parallel patterns, or relationship pattern*, which shows how 'similarities of decisions' are related to each other. Effects found are such as conditional relationship, where one parallel pattern has to happen before the next one occurs. Other effects includes associative, sequential and loop pattern effects. Our performance evaluation reports how relationship pattern performs in real life dataset and synthetic dataset and discusses some potential implementation issues.

1 Introduction

With the penetrate rate [1] of mobile technologies into the consumer market industry, many consumers are using mobile equipments in different situations in their life. Since mobile users usually carries mobile equipments most of the time, and the mobile equipment stays in touch with the mobile network access point at all times, source data that encompasses the behaviour of mobile users could be captured. Capturing this large amount of source data helps producing quality knowledge [11, 13, 15].

Mobile user data mining [11-15, 24] is a domain of data mining which emphasizes on developing methods into finding useful and interesting patterns from source data generated by mobile users. Data mining [3, 4] has been studied in many domains such as time series environment [5, 16-18], web based environment [6, 9, 10], geographical based environment [7, 8, 21, 22, 25] and location dependent environment [12, 20, 22, 23]. Enhancements in the area of data mining in general are also being developed, such as security enhancements [27] and efficiency in rules generation enhancements [26, 28, 29].

Mobile user data mining [12, 15, 22, 24] focuses extensively on mobile users and the mobile equipment that they carry. Parallel pattern [13, 15] is our previously

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 735 – 744, 2005.
© IFIP International Federation for Information Processing 2005

proposed method designed to find out similarities of decisions made by mobile users. In a mobile environment, there is a pattern that exists whereby similarities of decisions are taken among mobile users within a short period of time. The objective of this paper is to overcome the shortcomings of the previous proposed parallel pattern that only shows only one way relationship of similar change of states. In the interest of space, details on mobile user data mining is not provided.

2 Background

Parallel pattern [13, 15] finds out similarities of decision such as physical movements, or access to contents from the web, among mobile users. The word parallel in the parallel pattern reflects the event that happens at the same direction at the same time, such as two separation events from mobile user 1 and mobile user 2 to go from a fashion shop to a grocery shop within a specified period of time. The pattern in the word parallel pattern reflects the component to find out the pattern trend. If there is a pattern, 1) it is more likely to have reasons behind it 2) the pattern is likely to be repeated in the future. Therefore the study of parallel pattern provides a framework for understanding of mobile environment.

Definition 1 (General): The general definition of events, time, locations and mobile users in the mobile environment is as follows.

Let $E = \{e_1, e_2, \ldots, e_n\}$ be the set of events.
Let $T = \{t_1, t_2, \ldots, t_n\}$ be the set of time.
Let $L = \{l_1, l_2, \ldots, l_n\}$ be the set of location or theme.
Let $M = \{m_1, m_2, \ldots, m_n\}$ be the set of mobile users.

Definition 2 (Event): The mobile environment is a space over a time series where mobile users move freely as they wish and intends. Each event is recorded with the time and the mobile user identification and their movement from and movement to in a single event.

$$\sum_{time=0}^{time=n} [\sum_{m=0}^{m=\max} visits(from, to)]$$

Definition 3 (Window): The window is a specified timeframe, from *time_start* to *time_end* where events within this window are treated as occurred within the same window. Two events happening in two different window is considered as remote to each other such that: $E\{M, L(from, to), T]$, where $T >$ *time_start* and $T <$ *time_end.*

Definition 4 (Parallel Pattern): The parallel pattern is a trend evidenced by a higher than threshold volume of similar parallel activities occurring within the same window period of time such that: Parallel Pattern = $E(from, to)$, where $T >$ *time_start* and $T <$ *time_end* and $\forall E\ (from, to) : E_x(from, to) = E_y(from, to)$.

There are two types of parallel pattern [13, 15], namely *physical parallel pattern* and *logical parallel pattern*. The principle behind both is the same but the application environments among them are different. Physical parallel pattern looks at the physical location that mobile user travels, which are identified by means of geographical coordinates. Logical parallel pattern, on the other hand, looks at how the current logical state of the mobile user. This can be described as such that mobile user being currently in a shopping centre, even when some are in a shopping centre in the north of the city, and some are in a shopping centre in the south of the city. Logical parallel pattern treats them the same.

3 Proposed Method: Relationship Among Parallel Patterns

Relationship parallel pattern is our proposed method, which examines the relationships among parallel patterns – relationship patterns. There are four kinds of relationships among parallel patterns in a static node. These four relationships are namely *conditional*, *associative*, and *sequential* and *loop* relationship among parallel patterns.

3.1 Conditional Relationship

Definition 5 (Conditional Relationship): Let P be a list of parallel patterns found from a mobile user data mining episode. Let C be the condition for which two parallel patterns exists. Let $P = \{p_1, p_2, p_3, \ldots, p_n\}$. Let $C = \{c_1, c_2, c_3, \ldots, c_n\}$, where P is a list of parallel patterns, and C is a list of conditions. Let C_{ref} to be the reference condition which determines the existence of the relationship. Conditional relationship for two parallel patterns p_x, p_y is such that: C_n (p_1, p_2). Where $C_n = C_{ref.}$ Consider there is a conditional parallel pattern, $P1$ being Parallel Pattern 1 and $P2$ being Parallel Pattern 2, {If $P1$ = True Then $P2$} in the sample which consists of {$P1$, $P2$, $P3$, $P4$}. $P1$ must occur first before any $P2$ occurrence can occur. Once $P2$ have occurred, and if $P1$ occur after $P2$, then the statement no longer holds true. On the other hand, $P2$ must not occur before $P1$ have occurred. Finally, the statement must satisfy the confidence requirement, that is, must be sufficient frequent enough within the window sample in order to be qualified as a conditional parallel pattern.

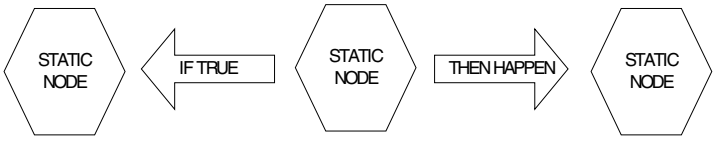


Fig. 1. Conditional Relationship among Parallel Patterns

Figure 1 shows the diagrammatic representation of conditional parallel pattern. Each arrow represents a parallel pattern found from previous data mining exercise, and all parallel patterns were drawn from within a static node in the centre. In the diagram, the left arrow shows a *TRUE* sign and the right arrow shows a *THEN HAPPEN* signs. It represents a conditional relationship in between these two parallel

patterns. For example, consider the parallel pattern on the left is $P1$ and the parallel pattern on the right is $P2$. If the algorithm finds out at most of the time, $P2$ only happens after $P1$ have occurred, then it can determine the confidence of conditional relationship among these two parallel patterns. Conditional parallel pattern occurs when one parallel pattern B occurs only after parallel pattern A occurred. It is not a conditional parallel pattern if there are instances in the window where parallel pattern B occurred before the occurrence of parallel pattern A. Therefore, this relationship must hold true at all times.

Figure 2 shows the algorithm for conditional parallel pattern. The algorithm first parses the list of source data and checks one set of parallel patterns which consists of two parallel patterns. Conditional parallel pattern is found when Parallel Pattern A always occurs with Parallel Pattern B within the same or immediately after the same time point, and that Parallel Pattern B never exist without the existence of Parallel Pattern A. When both of the conditions are satisfied, the two parallel patterns are marked as conditional parallel pattern.

```
Boolean Conditional Parallel Pattern (Pattern 1, Pattern 2) {
    For I = 0 to countPatterns (Parallel Patterns) Do
        Boolean Result = False;
        For J = 0 to countPatterns (Parallel Patterns) Do
            X = Parallel Pattern 1;
            Y = Parallel Pattern 2;
            If (X < Y) && (X !> Y) Then Result = True;
            Else Result = False;
    Return Result;
}
```

Fig. 2. Conditional Parallel Pattern

3.2 Associative Relationship

Definition 6 (Associative Relationship): Let T be a time sequence of $\{t_1, t_2, .., t_n\}$. Let P be a list of parallel pattern $\{p_1, p_2, .., p_n\}$. Consider the following list of input $\{t_1=p_1, t_2=p_1, p_2, t_3=p_3, t_4=p_4\}$ and the unit of buffer is 1 time unit. Therefore, since p_1 and p_2 occurred at t_2 itself, it qualifies as an associative parallel pattern because they both occurred at a precise time point. On the other hand, the fact that p_1 and p_2 occurred at t_1 and t_2 are considered relatively near as the time buffer is 1 unit, which is exactly what is happening right now. Therefore, the above fact also qualifies p_1 and p_2 as an associative parallel pattern: (E_x, E_y) : $time(E_x) = time(E_y)$, and $frequency$ $(E_x, E_y) \geq frequency_threshold$.

Figure 3 shows the diagrammatic representation of associative parallel pattern. It can be observed that during time 1, two parallel patterns occurred at the same time. During time 2, there are also two parallel patterns occurred at the same time. Figure 4 shows the algorithm for mining associative parallel pattern. The algorithm first receives the list of transactions and the conditions specified by the user. Conditional parallel pattern is designed to find out frequent instances that two parallel pattern occurring within the relatively same period of time, in this case, ± 5 seconds. Once they are found, they are marked as associative parallel pattern. Associative properties occur when two parallel patterns occurring at the same time. As the chances of two

parallel patterns happening at precisely the same time, such as precision up to 1 millisecond is very rare, therefore, it should be adjusted so that parallel patterns that are happening at somewhere nearby timeframe will prevent the situation of too much precision returning too little result.

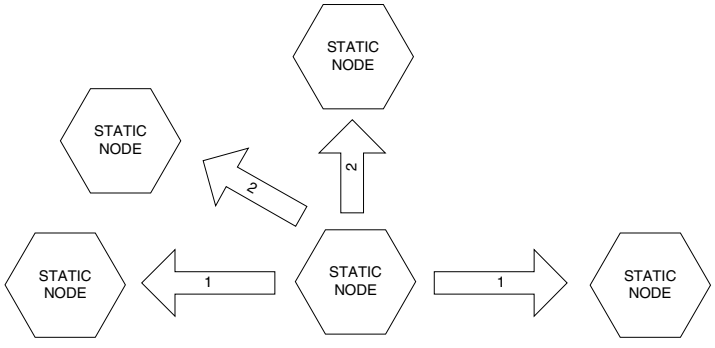


Fig. 3. Associative Parallel Pattern

```
Boolean Associative Parallel Pattern (Parallel Pattern List) {
    Boolean Result = False;
    For I = 0 to countSequence(Parallel Pattern List) Do
        If (countFrequency(I) > freq_threshold) Then
            List(I).Status = Frequent;
        Else List(I).Status = Infrequent;
    For All List.Status = Frequent Do
        L = formCombination(attribute_length);
        L.frequency = countFrequency(L, List);
        If (L.frequency > support) Then Result = True;
        Else Result = False;
 }
```

Fig. 4. Associative Parallel Pattern

3.3 Sequential Relationship

Definition 7 (Sequence Relationship): The sequence relationship pattern $\{e_1, e_2, e_3, e_4\}$ represent a sequence of events, each item being an event, such that one event happens one after the previous event. In this case, $e_1 \rightarrow e_2 \rightarrow e_3 \rightarrow e_4$. In this case, $e1$ depends on nothing, e_2 depends on e_1, e_3 depends on e_2 and e_1, e_4 depends on e_3, e_2 and e_1. Therefore, sequential parallel pattern can be seen as a sequence of conditional parallel pattern. Figure 5 illustrates: $frequency(E_{time=1} \rightarrow E_{time=2} \rightarrow E_{time=3} \rightarrow E_{time=4}) \geq frequency_threshold$.

Figure 6 shows the algorithm for mining sequential parallel pattern. The algorithm first receives the list of parallel pattern and the confidence threshold. After receiving these inputs, the list of transactions in the windows are examined by matching each pass of window with a sequence of possible sequential parallel pattern. The frequency of occurrence for each sequence are then recorded and divided by the window size to

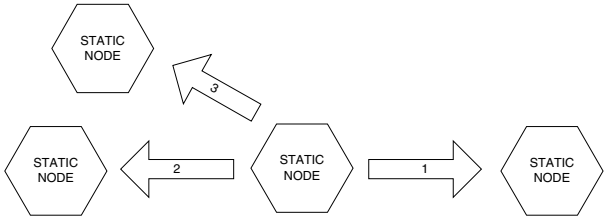


Fig. 5. Sequential Parallel Pattern

show the confidence of the sequence being a sequential parallel pattern. If the condition are satisfied, that is the sequence occurs frequent enough to be considered high confidence, then the sequence will be accepted as a sequential parallel pattern.

```
Boolean Sequential Parallel Pattern (Parallel Pattern List) {
    For I = 0 to formSequence(Parallel Pattern List) Do
        Boolean Result = False;
        SQ = currentSequence();
        If countOccurence(SQ) > sequence_threshold) Then
            If (last(SQ) = nextFirst(SQ) - 1) Then Result = False;
            Result = True;
        Else Result = False;
    Return Result;
}
```

Fig. 6. Sequential Parallel Pattern

3.4 Loop Relationship

Definition 8 (Loop Relationship): Consider the sequential parallel pattern of $\{p_1, p_2, p_3, p_4, p_5\}$. It qualifies as a loop parallel pattern when p_1 occurs after p_5 have occurred and p_1, p_2, p_3, p_4 and p_5 follows a sequence relationship, that is, one occurs after the previous.

Let S be a list of sequence. S_{ref} is the reference sequence in which it has occurred frequently to be considered a sequential relationship pattern. Loop Relationship = $time1(S_{ref}) \rightarrow time2(S_{ref})$ By finding the circular relationships among parallel patterns, it can be said that each circular relationship refers to a phenomenon. A phenomenon is something reoccurs at various frequencies, but the whole cycle do repeats itself. Of course, the window size must be large enough to accommodate the possible loop size of the circular flow. Furthermore, there must exist for a few repetition of the circular relationship in order to suggest a high confidence of the statement.

Figure 7 shows the diagrammatic representation of loop parallel pattern. It can be seen that there are two sets of parallel pattern that occurs over the same period amount of time, namely time series {1, 3, 5, 7} and {2, 4, 6, 8}. Figure 8 shows the algorithm to find the loop characteristics among the parallel patterns. The loop characteristic is an extension of multiple reoccurrence of sequential parallel pattern without any break. If a loop parallel pattern is found, it shows that there is a highly significant

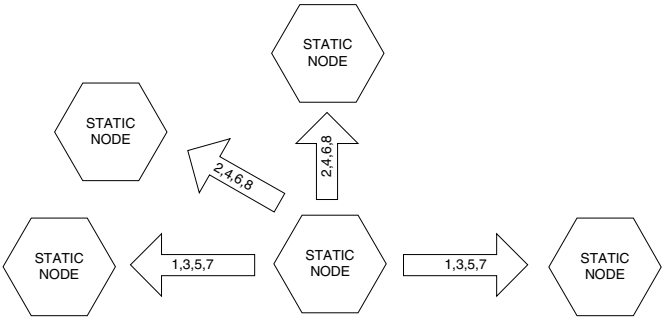


Fig. 7. Loop Parallel Pattern

```
Boolean Loop Parallel Pattern (Parallel Pattern List) {
    For I = 0 to formSequence(Parallel Pattern List) Do {
        Boolean Result = False;
        SQ = currentSequence();
        If countOccurence(SQ) > sequence_threshold) Then {
            If (last(SQ) = nextFirst(SQ) – 1) Then Result = True;
            Else Result = False;
        Else Result = False;
    Return Result;
}
```

Fig. 8. Loop Parallel Pattern

reoccurrence of patterns, in which during random events are highly unlikely. However, this algorithm is written to preserve the need in case this actually occurs so that it could be detected.

4 Performance Evaluation

Performance evaluation is performed under an IBM PC Pentium IV 384MB memory environment. Dataset used are *R*, *R3* and *R5* derived from dataset generator, with random seed derived from generation of integers with repetition but fixed within a specified range. The objective is to examine the relationship among the window size

Variable	Description
window	Size of window used to perform data mining.
frequency	Frequency of occurrence for a certain pattern.

Dataset	Description
R	Fully random dataset.
R3	Sequence repeated after 3 units of time interval.
R5	Sequence repeated after 5 units of time interval.

Fig. 9. Description for R, R3 and R5 Datasets

towards the responsiveness of finding relationships among parallel pattern. Therefore, data for *window size* 10 is towards *time* 10, *window size* 20 is towards *time* 20, *window size* 30 is towards *time* 30. Random source data are sourced from *random.org* [19]. The randomness is sourced from atmospheric noise and fed into the seed for random number generator.

Figure 9 shows the description of variables involved in the performance evaluation and description of individual dataset used for performance evaluation, namely {*R*, *R*3 and *R*5}. Due to lack of real life dataset, performance evaluation is conducted using synthetic dataset. These synthetic dataset are obtained from a random number generator with the seed being replaced via atmospheric noise. *R*, also known as *Random* or *R*1 is a truly random dataset, and it sets as a baseline of a data where no pattern can be found as the whole dataset is random. *R*3 and *R*5 are repeated dataset, where *R*3 have a repeat every 3 units of time interval and *R*5 have a repeat every 5 units of interval. The reason for doing so is to create some form of artificial patterns, so that the algorithms can extract the patterns out.

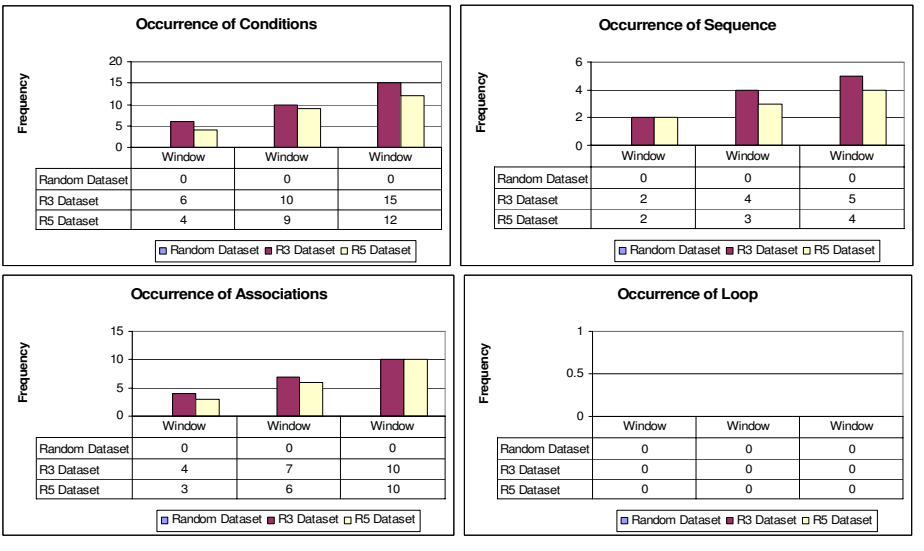


Fig. 10. Performance Result using R, R3 and R5 Datasets

Figure 10 shows the performance result for the 4 different mining algorithms under the 3 different dataset configuration under 3 different window sizes. For conditional relationship pattern, it was observed that during the course of testing Dataset *R*3 always returns higher result than Dataset *R*5. For sequential relationship pattern, it was observed that the trend of graph is growing gradually, and this is caused by increasing window size in the *x* axis. *R*3 is always higher than *R*5 because there are more complete repetitions in *R*3 than *R*5 dataset. For associative relationship pattern, it was observed that the reading is growing up because window size is increasing in the *x* axis. *R*3 and *R*5 happen to be equal number of reading because of coincidence. For loop relationship pattern, it was observed that no loop parallel pattern is found, due to its rare occurrence.

5 Conclusion and Future Work

This paper reports our work on finding relationships among parallel pattern - relationship pattern. *Conditional relationship* represents a movement patterns happening one after another such as if users move from *A* to *B*, then it is likely that users will move from *C* to *D* as well. *Sequence relationship* represents a movement patterns happening in sequence such as if users move from *A* to *B*, then it is likely for users to move from *C* to *D*, and the it is likely that users will move from *E* to *F* after this. *Association relationship* represents movement patterns happening at the same time such as users will move from *A* to *B* or *C* to *D* at the same time. *Loop relationship* represents patterns that repeat itself once finished and it is a sequential relationship where the last occurrence of pattern connects back to the first occurrence of pattern, such as users will move to *A* to *B*, then *C* to *D*, then *E* to *F*, and then pattern of users moving from *A* to *B* will occur again.

It represents a progress towards a higher level of mobile user data mining from the previously found patterns in the lower level. With higher level of knowledge being found, higher level knowledge represents knowledge that are rich and simplified as higher level knowledge is extracted based on the summary of previously found knowledge. It has been concluded that the ability for the proposed method being able to find knowledge at a higher level, it is recommended for mobile user data mining. Our future work for this research area is to extend the concept of high level quality knowledge mining over other areas such as *frequency pattern* [11], *group pattern* [24], *association rules* [3], and *sequential patterns* [4].

References

1. Statistics for Telecom Services for2004. http://www.ida.gov.sg/idaweb/media/infopage.jsp?infopagecategory=factsheet:factfigure&versionid=1&infopageid=I2703 Accessed: 20 June 2004
2. University of California (Irvine) Library - Knowledge Discovery in Database Archive. http://kdd.ics.uci.edu/ Accessed: 1st Dec 2004
3. R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In Proc. *20th International Conference on Very Large Data Bases*, pp. 487-499, 1994.
4. R. Agrawal and R. Srikant. Mining Sequential Patterns. In Proc. *11th International Conference on Data Engineering*, pp. 3-14, 1995.
5. S. Y. Chen, X. Liu, Data mining from 1994 to 2004: an application-oriented review, *Int Journal of Business Intelligence and Data Mining*, vol. 1, no. 1, pp. 4-21, 2005.
6. V. Christophides, G. Karvounarakis, and D. Plexousakis. Optimizing Taxanomic Semantic Web Queries using Labeling Schemes. *Journal of Web Semantics*, **1**(2):207-228, 2003.
7. I. Claude, J.-L. Daire, and G. Sebag. Fetal Brain MRI: Segmentation and Biometric Analysis of the Posterior Fossa. *IEEE Transactions on Biomedical Engineering*, **51**(4):617-626, 2004.
8. D. E. Cooper, P. Ezhilchelvan, and I. Mitrani. High Coverage Broadcasting for Mobile Ad Hoc Networks. *Lecture Notes in Computer Science,* vol. 3042, pp. 100-111, 2004.
9. D. L. Lee, M. Zhu, H. Hu, When location-based services meet databases, *Mobile Information Systems*, vol. 1, no. 2, pp. 81-90., 2005.
10. M. Eirinaki and M. Vazirgaiannis. Web Mining for Web Personalization. *ACM Transactions on Internet Technology*, vol. 3, no. 1, pp. 1-27, 2003.

11. Goh, J and Taniar, D., Mining Frequency Pattern from Mobile Users. *Knowledge-Based Intelligent Information & Engineering and Systems, Lecture Notes in Computer Science Part III,* vol. 3215, pp. 795-801, 2004.
12. Goh, J and Taniar, D., Mobile Data Mining by Location Dependencies. *5th International Conference on Intelligent Data Engineering and Automated Learning, Lecture Notes in Computer Science,* vol. 3177, pp. 225-231, 2004.
13. Goh, J and Taniar, D., Mining Physical Parallel Pattern from Mobile Users. *International Conference on Embedded and Ubiquitous Computing, Lecture Notes in Computer Science,* vol. 3207, pp. 324-332, 2004.
14. Goh, J and Taniar, D., An Efficient Mobile Data Mining Model. *2nd International Symposium on Parallel and Distributed Processing and Applications, Lecture Notes in Computer Science,* vol. 3358, pp. 54-59, 2004.
15. Goh, J and Taniar, D., Mining Parallel Pattern from Mobile Users. *International Journal of Business Data Communications and Networking,* vol. 1, pp. 50-76, Jan - Mar 2005.
16. J. Han, G. Dong, and Y. Yin. Efficient Mining of Partial Periodic Patterns in Time Series Database. In Proc. *International Conference on Data Engineering*, pp. 106-115, 1999.
17. J. Han, W. Gong, and Y. Yin. Mining Segment-Wise Periodic Patterns in Time Related Databases. In Proc. *In Proc. 4th Int. Conf. on Knowledge Discovery and Data Mining*, pp. 214-218, 1998.
18. J. Han, J. Pei, and Y. Yin. Mining Frequent Patterns without Candidate Generation. In Proc. *In Proc. Int. Conf. Special Interest Group of Management of Data*, pp. 1-12, 2000.
19. M. Harr. Random.org. http://www.random.org Accessed: 1st August 2004
20. G. Kastaniotis, N. Zacharis, T. Panayiotopoulos, and C. Douligeris. Intelligent Web Prefetching Based upon User Profiles - The WebNaut Case. *Lecture Notes in Artificial Intelligence,* vol. 3025, pp. 54-62, 2004.
21. K. Koperski and J. Han. Discovery of Spatial Association Rules in Geographical Information Databases. In Proc. *4th International Symposium on Advances in Spatial Databases*, pp. 47-66, 1995.
22. D. L. Lee, J. Xu, B. Zheng, and W.-C. Lee. Data management in location-dependent information services. *Pervasive Computing, IEEE*, vol. 1, no. 3, pp. 65-72, 2002.
23. J. Li, B. Tang, and N. Cercone. Applying Association Rules for Interesting Recommendations Using Rule Templates. *8th Pacific-Asia Conference on Knowledge Discovery and Data Mining 2004, Lecture Notes in Artificial Intelligence,* vol. 3056, pp. 166-170, 2004.
24. E.-P. Lim, Y. Wang, K.-L. Ong, and et al. In Search of Knowledge About Mobile Users. *ERCIM News*, vol. 1, no. 54, pp. 10, 2003.
25. W.-C. Lin, D.-Y. Liao, C.-Y. Liu, and Y.-Y. Lee. Daily Imaging Scheduling of an Earth Observation Satellite. *Systems, Man and Cybernetics, Part A, IEEE Transactions on*, vol. 35, no. 2, pp. 213-223, 2005.
26. T. Miyahara, Y. Suzuki, T. Shoudai, T. Uchida, K. Takahashi, and H. Ueda1. Discovery of Maximally Frequent Tag Tree Patterns with Contractible Variables from Semistructured Documents. *8th Pacific-Asia Conference on Knowledge Discovery and Data Mining 2004, Lecture Notes in Artificial Intelligence,* vol. 3056, pp. 133-144, 2004.
27. S. R. M. Oliveira, O. R. Zaiane, and Y. Saygin. Secure Association Rule Sharing. *8th Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2004, Lecture Notes in Artificial Intelligence,* vol. 3056, pp. 74-85, 2004.
28. H. C. Tjioe, D. Taniar, Mining Association Rules in Data Warehouses, *Intl. J. of Data Warehousing and Mining*, vol. 1, no. 3, pp. 28-62.
29. P. K. C. Tse, W. K. Lam, K. W. Ng, C. Chan, An Implementation of Location-Aware Multimedia Information Download to Mobile System, *Journal of Mobile Multimedia*, vol. 1, no. 1, pp. 33-46.

Asymmetry-Aware Link Quality Services in Wireless Sensor Networks

Junzhao Du[1,2], Weisong Shi[1], and Kewei Sha[1]

[1] Department of Computer Science,
Wayne State University, Detroit, MI 48202, USA
[2] Software College,
Xidian University, Xi'an, Shaanxi 710071, P.R. China

Abstract. Recent studies in wireless sensor networks (WSN) have observed that the irregular link quality is a common phenomenon, rather than an anomaly. The irregular link quality, especially link asymmetry, has significant impacts on the design of WSN protocols. In this paper, we propose two asymmetry-aware link quality services: *the neighborhood link quality service (NLQS)* and *the link relay service (LRS)*. The novelty of the NLQS service is taking the link asymmetry into consideration to provide timeliness link quality and distinguishing the inbound and outbound neighbors with the support of LRS, which builds a relay framework to alleviate the effects of link asymmetry. To demonstrate the proposed link quality services, we design and implement two example applications, *the shortest hops routing tree (SHRT)* and *the best path reliability routing tree (BRRT)*, on the TinyOS platform. We found that the performance of two example applications is improved substantially. More than 40% of nodes identify more outbound neighbors and the percentage of increased outbound neighbors is between 14% and 100%. In SHRT, more than 15% of nodes reduce hops of the routing tree and the percentage of reduced hops is between 14% and 100%. In BRRT, more than 16% of nodes improve the path reliability of the routing tree and the percentage of the improved path reliability is between 2% to 50%.

1 Introduction

Recent empirical studies [1,2,3,4,5,6] on Berkeley motes platform show that there are highly irregular links in real deployments of wireless sensor networks (WSN). The packet delivery performance varies significantly with spatial and temporal factors and it is difficult to get the timeliness link quality information between neighbors. Furthermore, approximately 5% to 15% of all links are asymmetric links and asymmetric links vary significantly in different directions and distances. Asymmetric links, especially unidirectional links, bring the problem to the design of WSN protocols, such as MAC protocols, neighborhood and topology discovery protocols, and routing protocols.

Motivated by the recent studies on link irregularity, we intend to alleviate the problem caused by asymmetry on link quality services. On the other hand, we observed an interesting phenomenon that one or two steps relay is enough to forward packets across unidirectional links. This inspires us to design a relay service to utilize those otherwise useless asymmetric unidirectional links in traditional wisdom. To this end,

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 745–754, 2005.
© IFIP International Federation for Information Processing 2005

we have designed and implemented asymmetry-aware link quality services, including *the neighborhood link quality service (NLQS)* and *the link relay service (LRS)*, to provide the timeliness link quality information of neighbors and build a relay framework to alleviate effects of the link asymmetry on the TinyOS platform [7]. We leverage the Window Mean Exponentially Weighted Moving Average Estimator (WMEWMA) [3,8] to estimate the packet reception rate (PRR) between neighbors. Our proposed link quality services are able to identify more outbound neighbors and provide timeliness link quality information of inbound and outbound neighbors. Both of these services are crucial for the design of link quality aware routing protocols. Our services also provide a relay framework to forward packets across unidirectional links, which have more positive contributions to localized coordination between neighbors. To demonstrate our proposed link quality services, we design and implement two example applications, building *the shortest hops routing tree (SHRT)* and *the best path reliability routing tree (BRRT)*. To evaluate our proposed link quality services, we have conducted both static analysis and simulation using the TOSSIM simulator [9]. We found that the performance of two example applications was improved substantially.

The rest of the paper is organized as follows. The design and implementation of the link quality services are presented in Section 2. SHRT and BRRT are described in Section 3. In Section 4, performance evaluation results are discussed. Related work and concluding remarks are listed in Section 5 and 6 respectively.

2 Asymmetry-Aware Link Quality Services

We consider the topology of a WSN as a weighted directed graph (G). All motes in the topology are vertexes of the graph and all vertexes belong to the vertex set, called $V(G)$. In this paper, we also call the vertex of a graph as node. All links in the topology are edges of the graph and all edges belong to the edge set, called $E(G)$. We use $e(u, v)$ to represent the edge from Node u to Node v, and assign the estimated PRR of the link as $weight$ of the edge. Figure 1(a) is an example topology. In this topology, every node has some inbound neighbors and outbound neighbors. In a WSN, a node can identify inbound neighbors only when it receives some packets from them. To identify outbound neighbors, the node should get acknowledgement from them. For example, Node 8 and Node 1 can receive packets from each other, then one of them will identify its counterpart as inbound and outbound neighbor. However, the link quality, in term of PRR, from Node 8 to Node 1 is different from that from Node 1 to Node 8, and the PRR will vary with time. Due to the asymmetric links, node can not identify all of its outbound neighbors. For example, Node 22 can identify Node 8 as its inbound neighbor. But Node 8 can not directly identify Node 22 as its outbound neighbor. Nevertheless, if Node 9 can relay the acknowledgement for Node 22, so that Node 8 will identify Node 22 as its outbound neighbors. In some situations, one step relay is not enough to identify all outbound neighbors. For example, if Node 24 will identify Node 9 as its outbound neighbor, both Node 23 and Node 52 should relay the acknowledgement from Node 9 to Node 24. The more steps the acknowledgement is relayed, the more the outbound neighbors will be identified. Next, we give a formal definition of *theoretical outbound neighbors*, *No Relay outbound neighbors*, *One-Step Relay outbound neighbors*, and *Two-Step Relay outbound neighbors*.

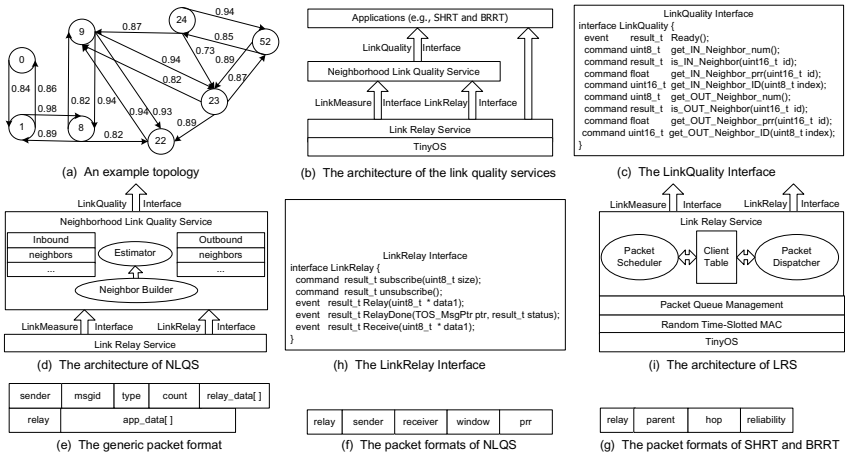


Fig. 1. The figures of the link quality services

Definition 1. *Theoretical outbound neighbors of node v (N(v)):* $\{u|\forall u \in V(G) \wedge e(v,u) \in E(G)\}$

Definition 2. *No Relay outbound neighbors of node v (N0(v)):* $\{u|\forall u \in V(G) \wedge e(v,u) \in E(G) \wedge e(u,v) \in E(G)\}$

Definition 3. *One-Step Relay outbound neighbors of node v (N1(v)):* $\{u|(u \in N0(v)) \vee (\forall u \in V(G) \wedge e(v,u) \in E(G) \wedge \exists m \in V(G) \wedge e(u,m), e(m,v) \in E(G))\}$

Definition 4. *Two-Step Relay outbound neighbors of node v (N2(v)):* $\{u|(u \in N1(v)) \vee (\forall u \in V(G) \wedge e(v,u) \in E(G) \wedge \exists m,n \in V(G) \wedge e(u,m), e(m,n), e(n,v) \in E(G))\}$

Continuing with the above example, $N(24)$ includes Node 9, Node 23 and Node 52; $N0(24)$ only includes Node 52; $N1(24)$ includes Node 23 and Node 52; $N2(24)$ includes Node 9, Node 23, and Node 52. From these definitions and the above example, we know that we might find a relay path to relay packets across unidirectional links. If there is a path, there will be opportunity to relay packets across unidirectional links. If there is only one node in the middle of the path, we call this path *One-Relay path*. If there are two nodes in the middle of the path, we call this path *Two-Relay path* and so on. In next section, we will describe the link quality services, which can build inbound and outbound neighbors with timelessness link quality information and relay packets across unidirectional links.

2.1 System Overview

Next, we describe the design and implementation the link quality services, including NLQS and LRS, on the TinyOS platform. In NLQS, we measure and estimate the link quality using link quality estimator, build the inbound and outbound neighbors tables, and provide the *LinkQuality* interface for applications to query the timeliness link quality information of neighbors. In LRS, we propose a link relay protocol, implement a link

relay framework, and provide the *LinkRelay* interface for applications to relay packets across unidirectional links. Furthermore, NLQS uses the *LinkRelay* interface to relay the estimated PRR of the inbound neighbors back to them. Figure 1(b) shows the architecture of the link quality services. Due to space limitation, we provide the high level view only, interested readers please refer the technical report version of this paper [10].

2.2 Neighborhood Link Quality Service

NLQS provides the *LinkQuality* interface. Figure 1(c) lists the interface, which consists of one event and eight commands, complying with the TinyOS and nesC language [7]. When the service is ready, it will signal the $Ready$ event to notify the availability of the service. Even the service is ready for use, NLQS will continue measure and estimate the timeliness link quality between neighbors and update inbound and outbound neighbors tables. Applications can call those commands to query the link quality information. The link quality information includes that how many inbound and outbound neighbors are identified, what are their node ID, what are the estimated PRR from inbound neighbors and to outbound neighbors.

To provides this service, we need to measure and estimate the link quality between neighbors and build the inbound and outbound neighbors tables with the estimated PRR. To do this, every node will periodically broadcast some packets, which contain node ID and packet ID. Other nodes overhear the channel for the broadcast packets to measure and estimate the link quality from their inbound neighbors. Every node only measures and estimates PRR of the link from its inbound neighbors. We leverage WMEWMA to estimate PRR based on current and history measured results. WMEWMA uses a time window to observe the received packets and it adjusts the estimation result using latest average value of the measured PRR. When nodes have estimated PRR for their inbound neighbors, they will send the estimated results back to those inbound neighbors. When their inbound neighbors get the estimated results from their outbound neighbors, they can identify those outbound neighbors and build the outbound neighbors table with the estimated results. As we have discussed before, due to the link asymmetry, nodes may not directly receive packets from their outbound neighbors. They will get the estimated PRR to their outbound neighbors from LRS (Section 2.3).

Figure 1(d) shows the architecture of NLQS, which consists of four components: the inbound neighbor table, the outbound neighbor table, the neighbor builder, and the link quality estimator. The neighbor builder builds the inbound neighbor table using the PRR estimated by the link quality estimator and builds the outbound neighbor table using the PRR getting from LRS.

2.3 Link Relay Service

LRS is a general service to relay packets across unidirectional links and it is an additional layer between TinyOS and applications. Every node provided LRS will collaborate with each other to forward relay packets. In LRS, we propose a link relay protocol for inter-node communication, and design a relay framework and provide an interface for applications to relay packets.

Figure 1(e) illustrates a generic packet format used by LRS for inter-node communication. The *type* field indicates the type of the relay data. In LRS, every *type* of the

relay data is associated with one application and LRS uses the value of the *type* field to dispatch the relay data to the associated application. The *relay_data* field contains the relay data, which will be forwarded to the application. Usually the first field of the relay data is the *relay* field and the value of this field indicates how many relay steps this relay data can be relayed. The value of the *relay* field will be decreased by one for every relay step until it reaches zero. If every node initializes the *relay* field as one, then the relay packet will be relayed once. We call this *One-Step Relay* algorithm. If every node initializes it as two, then the relay packet will be relayed twice. And we call this *Two-Step Relay* algorithm and so on. The more the packets are relayed, the more neighbors will receive the packets and the more opportunities exist to bypass unidirectional links. Applications will store its data in the *app_data* field. Figure 1(f) is the packet format used in NLQS to relay the estimated PRR to inbound neighbors (Section 2.2). Figure 1(g) is the packet format used in SHRT and BRRT (Section 3).

Figure 1(h) lists the *LinkRelay* interface. This interface is a parameterized interface, which can support 256 applications concurrently. As we have discussed that every application will be associated with an unique ID. This ID will be store in the $type$ field of the relay packet. This mechanism is similar to the active message on TinyOS platform. When an application will use LRS to relay packet, it first call the *subscribe* command. When a relay packet arrives this node, the associated application will get the *Receive* event from LRS. At this time, the application can make decision about how to process this incoming relay data. When the radio channel is available to send data, LRS will signal a *Relay* event to get the relay data from applications. At this time, the application can make decision about which relay data will be relayed and how many steps the relay data will be relayed. If the application is not interested in the LRS events any more, it can call the *unsubscribe* command to unsubscribe these events. In LRS, we follow the basic idea in system design to separate mechanism and policy . We design and implement the link relay mechanism and provide this interface for applications to make policy decisions.

Figure 1(i) shows the architecture of LRS. The packet dispatcher dispatches the relay data to applications. The packet scheduler schedules applications to forward relay data. The packet queue management manages packets received from the TinyOS platform and created by applications. The random time-slotted MAC is designed to avoid collisions of simultaneous packets transmissions between neighbors.

3 Example Applications

To demonstrate and evaluate our link quality services, we design and implement two example applications on the TinyOS platform: *the shortest hops routing tree* (SHRT) and *the best path reliability routing tree* (BRRT). In SHRT, every node has a routing path with shortest hops to the sink node. In BRRT, every node has a routing path with best reliability to the sink node. We define the path reliability for every node as follows: $Reliability_A = \prod_A^S PRR$, where the $Reliability_A$ is the path reliability for node A. The PRR is the packet reception rate of the links along the routing path from Node A to the sink node S. For example, in Figure 1(a), Node 0 is the sink node and the path reliability of Node 9 is $0.72 = (0.94 * 0.89 * 0.86)$.

In SHRT and BRRT, every node initializes its hop count as the maximum value or its path reliability as zero, and its parent as empty. The sink node floods a packet, which contains its node ID and its hop count or its path reliability. The hop count of sink node is 0 and path reliability is 1.0. When receiving the flooding packet, node will update its parent and hop count or path reliability according to the received packet. In SHRT, if the hop count of the node in the flooding packet is smaller than that of the current parent, the receiver will choose the node as its parent and update its hop count based on the new parent. In BRRT, the receiver will choose the node as its parent, based on which the node can get better path reliability than based on current parent. After node updates its parent, it will broadcast its routing tree information again. This process continues until no node updates its parent.

However, due to the link asymmetry, nodes have more chances to build a broken routing tree when they choose the parent only considering the information embedded in the received packet. For example, in Figure 1(a), if Node 22 receives a flooding packet from Node 8, Node 22 will choose Node 8 as its parent. Unfortunately, Node 22 will build a broken routing tree. To avoid this, nodes should leverage NLQS to check whether the node in the flooding packet is one of their outbound neighbors before choosing it as its parent. The link asymmetry also has negative effects on building a better routing tree. For example, in Figure 1(a), Node 9 is the best parent candidate for Node 24. But due to the unidirectional link, Node 24 can not get flooding packet from Node 9. In this topology, Node 24 can get flooding packet from Node 52 and will choose Node 52 as its parent. In this case, Node 24 needs more hops to route packets to the sink node. However, if Node 9 is able to relay its flooding packets across the unidirectional link using LRS, Node 24 can get the flooding packet and will choose Node 9 as its parent. Our technical report [10] has the detail algorithms to build the SHRT and BRRT using NLQS and LRS.

4 Performance Evaluation

We are now in the position to evaluate the performance of our link quality services. We propose and define the following performance metrics: *the Number of Increased Outbound Neighbors, the Number of Reduced Hops in SHRT, the Path Reliability Improvement in BRRT,* and *Energy Consumption in SHRT and BRRT.* Based on these performance metrics, we conduct a static analysis to calculate the optimal results in an ideal network without packet loss. Then we simulate our link quality services using TOSSIM [9] in the lossy model. We totally simulate 156 motes. The bit error rate of links are randomly chosen from the real measurements using Berkeley MICA2 motes in a controlled environment.

As we know in Section 2 and Section 3 that the more steps the packets are relayed, the more outbound neighbors will be identified in NLQS and the better routing tree will be built in SHRT and BRRT. So we will simulate *No Relay*, *One-Step Relay*, and *Two-Step Relay* algorithms in our link quality services. We hope to compare the benefit and the overhead of the link relay mechanism and relay steps. In SHRT and BRRT, due to the lossy links and collisions, the flooding packets have a high probability to be lost. To make sure most nodes can receive the flooding packets, we define three policies to forward it. First, the node only forwards its routing tree information twice *(P2 Policy)*.

Second, the node will forward its routing tree information twice and forward the relay information once *(P2+R1 Policy)*. Third, the node will forward its routing tree information and the relay information twice *(P2+R2 Policy)*. In all, we will simulate *No Relay*, *One-Step Relay*, and *Two-Step Relay* algorithms with *P2 Policy*, *P2+R1 Policy*, and *P2+R2 Policy*.

Increased Outbound Neighbors. Using our link quality services, node will identify more outbound neighbors. Figure 2 (a) shows the cumulative distribution function for the percentage of increased outbound neighbors using link quality services. In this figure, the x-axis stands for the percentage of increased outbound neighbors for every node and the y-axis stands for the percentage of these kinds of nodes. Results of the *One-Step Relay* and *Two-Step Relay* algorithms in the analysis and simulation are shown together to facilitate comparison. From this figure, we can see that more than 40% of nodes identify more outbound neighbors in the *One-Step Relay* algorithm and the percentage of increased outbound neighbors is from 17% to 100%. Also we can find that more than 45% of nodes identify more outbound neighbors in the *Two-Step Relay* algorithm and the percentage of increased outbound neighbors is from 14% to 100%. Note that the analysis results is a little better than those in the simulation. The reason for this is that some acknowledgement packets are lost due to loss links in the simulation. Therefore some nodes fail to identify those outbound neighbors. The *Two-Step Relay* algorithm is better than *One-Step Relay* as we expected.

Reduced Hops in SHRT. With the help of our link quality services, we expect to build better SHRT, in which every node has shorter hops to the sink node than without using our services. Figure 2 (b) reports the CDF of the percentage of reduced hops for every node. In this figure, the x-axis is the percentage of reduced hops for every node and the y-axis shows the percentage of those kinds of nodes. From this figure, we can see that the results in the simulation do not match well with those in the analysis. In the analysis, there are more than 30% of nodes reduce their hops to the sink node and the percentage of reduced hops is mainly between 20% to 40%. In the simulation results, there are only more than 15% of nodes that reduce their hops to the sink node and the percentage of reduced hops is mainly between 15% and 25%. The *P2+R2 Policy* of *Two-Step Relay* is the best algorithm and more than 20% of nodes reduce their hops to the sink node. While the *P2+R1 Policy* of *One-Step Relay* is the worst one of all algorithms and only more than 13% of nodes reduce their hops to the sink node.

Improved Path Reliability in BRRT. With the help of our link quality services, we expect to build better BRRT, in which every node has better path reliability to the sink node than without using our services. Figure 2 (c) reports the CDF of the percentage of the improved path reliability for every node in the simulated topology. Figure 2 (d) is the zooming of the results. In these figures, the x-axis stands for the percentage of the improved path reliability for every node and the y-axis stands for the percentage of those kinds of nodes. In the simulation, more than 15% of nodes improve the path reliability and the percentage of the improved path reliability is mainly between 2% and 40% and the percentage of some nodes even reach on 50%. The combination of the *P2+R2 Policy* with the *Two-Step Relay* algorithm is the best one, where more than 27% of nodes improve the path reliability to the sink node and the percentage of the improved path reliability focus on 15% to 22% and 50%. The *P2 Policy* with the *One-*

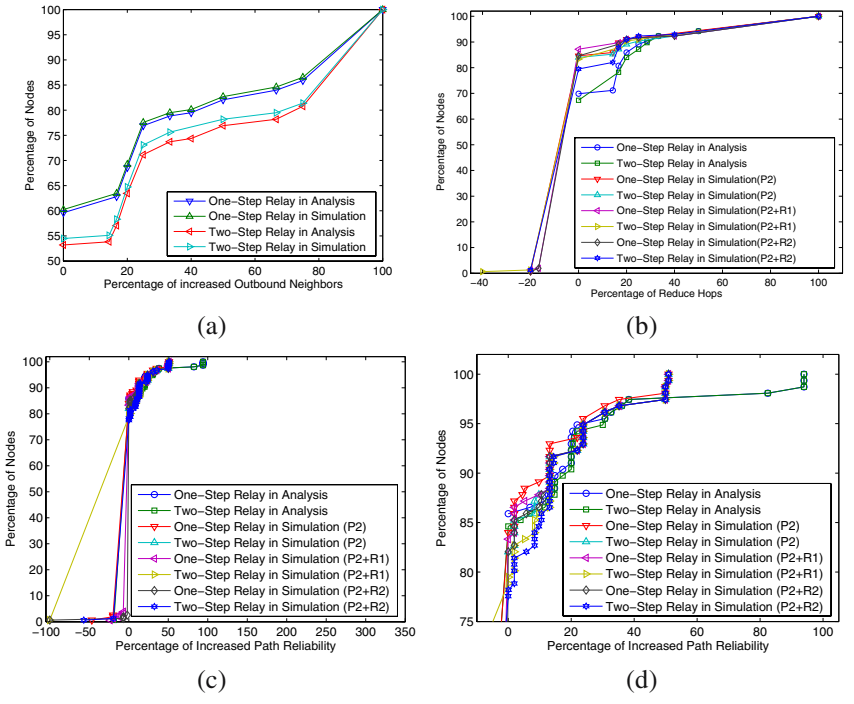


Fig. 2. Evaluation results: (a) The CDF of the percentage of increased outbound neighbors using link quality services, (b) The CDF of the percentage of reduced hops in SHRT, (c) The CDF of the improved path reliability in BRRT, and (d) Zooming of the CDF of the improved path reliability in BRRT

Step Relay algorithm is the worst one of all algorithms. Also we note that there are a lot of nodes improve the path reliability than the number in SHRT. Nearly every neighbor of a node in BRRT has different path reliability while most neighbors of a node in SHRT have same hops to the sink nodes. Therefore, nodes in BRRT have more opportunities to change their parents than they do in SHRT.

Energy Consumption in SHRT and BRRT. Figure 3 (a) shows the CDF of more packets sent by the *One-Step Relay* and *Two-Step Relay* algorithms than *No Relay* in SHRT. In this figure, the x-axis stands for more packets send by every node and the y-axis reports the CDF. From this figure, we can see that all algorithms have little difference in terms of the number of extra packets sent. For the *One-Step Relay* and *Two-Step Relay* algorithms, about 90% of nodes do not send more packets and about 10% of nodes send less packets. Figure 3(b) shows the CDF of more packets received by the *One-Step Relay* and *Two-Step Relay* algorithms than *No Relay* one. In this figure, the x-axis stands for more packets received by every node and the y-axis reports the CDF. From this figure, we can see the following results. The *P2+R2 Policy* of *Two-Step Relay* receives more packets than others. The *P2+R1 Policy* of *Two-Step Relay* is the second worst. The *P2 Policy* of *One-Step Relay* and the *P2 Policy* of *Two-Step Relay* follow the similar pattern and they receive less packets than other algorithms. Except the *P2+R2*

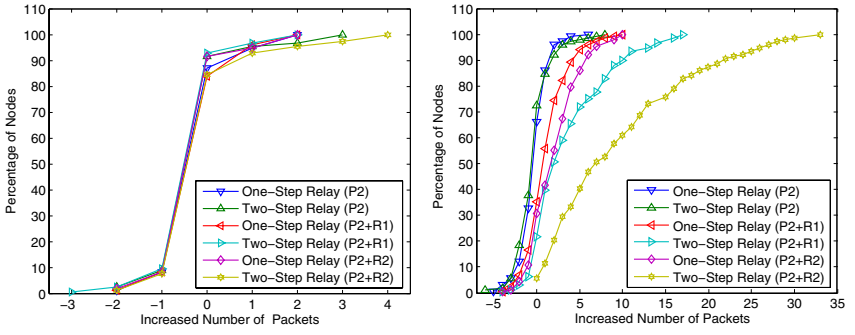


Fig. 3. (a) The CDF of more packets sent in SHRT, (f) The CDF of more packets received in SHRT

Policy of *Two-Step Relay*, most algorithms have nearly 20% of nodes which receive less packets than the *No Relay* algorithm. The CDF of more packets sent and received by the *One-Step Relay* and *Two-Step Relay* algorithms than *No Relay* one in BRRT have similar results with SHRT [10].

The results in this section show that there is a tradeoff between the energy consumption and the benefits brought by LRS. We argue that it is the application's decision to turn on/off LRS. For example, if the path reliability or path length is more important than the energy, e.g., event notification, then it should choose LRS. However, for monitoring-based applications it's better to turn off this service.

5 Related Work and Discussions

Our work is inspired by a variety of previous work, including *link layer characterization*, *link quality estimation*, and *existing solutions to link irregularity*.

Link layer characterization. Ganesan *et al.* find long links, backward links, asymmetric links, stragglers and clustering in large scale of WSN [1]. Zhao *et al.* measure the spatial and temporal characteristics of packet delivery and find the gray area in which there are significant variabilities in packet delivery performance and the no-determining relationship between signal strength and packet delivery [4]. Zhou *et al.* show that radio irregularity is a common phenomenon, which arises from multiple factors, such as variance in RF sending power, and different path losses depending on the direction of radio signal propagation [5]. All of these research work shows us the reality of the link quality in WSN, which make us have a deeper understanding of the wireless communication in WSN. We are the first to provide a set of APIs for link quality services.

Link quality estimation. Woo *et al.* show that WMEWMA is a very effective estimator, in term of stability and agility [8]. Cerpa *et al.* present a statistical model of lossy links in WSN [11]. Zhou *et al.* establish the Radio Irregularity Model (RIM) for simulation [5]. Zuniga *et al.* intend to identify the causes of the transitional region, and a quantification of their influence [6]. Different from these previous work, we focus on the asymmetry-

aware link quality services that alleviate the effect of asymmetric links in the link quality measurement and estimation process, which is neglected in previous research.

Existing solutions to link irregularity. Woo *et al.* study reliable and cost-based routing protocol techniques [3]. Seada *et al.* present energy-efficient forwarding strategies for geographic routing in lossy WSN [2]. Ramasubramanian *et al.* provide a bidirectional abstraction of the unidirectional network to routing protocols in *ad hoc networks* [12]. They results also show that reception based forwarding strategies are more efficient than purely distance-based strategies in the lossy link. We step further to distinguish inbound and outbound neighbors to make good use of them for high level routing service, and to provide a set of link quality interfaces. We also provide LRS for relaying packets across unidirectional links.

6 Conclusions

In this paper, we design and implement two asymmetry-aware link quality services. The performance of the services is evaluated in the context of two example applications, SHRT and BRRT. The detail simulation results using TOSSIM show that the performance of the applications can be improved substantially using proposed services.

References

1. Ganesan, D., Krishnamachari, B., Woo, A., Culler, D., Estrin, D., Wicker, S.: Complex behavior at scale: An experimental study of low-power wireless sensor networks. Technical Report UCLA/CSD-TR-02-0013, University of California, Los Angeles (2002)
2. Seada, K., Zuniga, M., Helmy, A., Krishnamachari, B.: Energy-efficient forwarding strategies for geographic routing in lossy wireless sensor networks. In: Proceedings of SenSys '04. (2004)
3. Woo, A., Tong, T., Culler, D.: Taming the underlying challenges of reliable multihop routing in sensor networks. In: Proceedings of SenSys '03. (2003)
4. Zhao, J., Govindan, R.: Understanding packet delivery performance in dense wireless sensor networks. In: Proceedings of SenSys '03. (2003)
5. Zhou, G., He, T., Krishnamurthy, S., Stankovic, J.A.: Impact of radio irregularity on wireless sensor networks. In: Proceedings of MobiSys '04. (2004)
6. Zuniga, M., Krishnamachari, B.: Analyzing the transitional region in low power wireless links. In: Proceedings of SECON '04. (2004)
7. Levis, P., et al.: The emergence of networking abstractions and techniques in tinyos. In: Proceedings of NSDI '04. (2004)
8. Woo, A., Culler, D.: Evaluation of efficient link reliability estimators for low-power wireless networks. Technical Report UCB/CSD-03-1270, University of California, Berkeley (2003)
9. Levis, P., Lee, N., Welsh, M., Culler, D.: Tossim: Accurate and scalable simulation of entire tinyos applications. In: Proceedings of SenSys '03. (2003)
10. Du, J., Shi, W., Sha, K.: Asymmetry-aware link quality services in wireless sensor networks. Technical Report MIST-TR-2005-006, Wayne State University (2005)
11. Cerpa, A., Wong, J., Kuang, L., Potkonjak, M., Estrin, D.: Statistical model of lossy links in wireless sensor networks. Technical Report CENS Technical Report, University of California, Los Angeles (2004)
12. V. Ramasubramanian, R.C., Mosse, D.: Providing a bidirectional abstraction for unidirectional ad hoc networks. In: Proceedings of INFOCOM '02. (2002)

Incorporating Global Index with Data Placement Scheme for Multi Channels Mobile Broadcast Environment

Agustinus Borgy Waluyo[1], Bala Srinivasan[1], David Taniar[2], and Wenny Rahayu[3]

[1] Caulfield School of Information Technology, Monash University, Australia
{Agustinus.Borgy.Waluyo,
Bala.Srinivasan}@infotech.monash.edu.au
[2] Clayton School of Information Technology, Monash University, Australia
David.Taniar@infotech.monash.edu.au
[3] Department of Computer Science and Computer Engineering,
Latrobe University, Australia
w.rahayu@latrobe.edu.au

Abstract. A scalable information delivery services like data broadcasting scheme is definitely of great interest especially to deal with the exponential increase of mobile users. In this paper, we present Global Indexing with data placement scheme for multi broadcast channels environment, which aims to minimize query access time, tuning time and power consumption. This scheme considers index and data allocation over separate broadcast channel. Moreover, it concerns with single and multiple data items request. A simulation model is developed to analyze the performance of the proposed scheme. We compare the proposed scheme with conventional scheme. It is found that the proposed scheme provides substantially better performances in every aspect of evaluation.

1 Introduction

In recent years, the use of wireless technology has been growing at an unbelievable rate. Most people are now able to connect to the network using portable size wireless computers powered by batteries (e.g. PDAs) and to conduct their business activities anywhere and anytime. These portable computers communicate with a central stationary server via wireless channel, and this paradigm is known as mobile computing [2,7].

Mobile computing environment is facing inherent limitations particularly power, storage and bandwidth capacity. With power capacity, the life expectancy of a battery (e.g. nickel-cadmium, lithium ion) was estimated to increase time of effective use by only another 15% [8]. Thus, efficient use of energy is definitely one of the main issues. Having considered the limitations of mobile environment, it is very much of benefit to incorporate a scalable and energy efficient mechanism for retrieving database information. Data dissemination (or also called broadcast mechanism) is considered the most appropriate strategy to deliver such services. Wireless broadcasting has been applied for decades in radio and television industry and it is very effective to cope with a massive number of clients. Two factors need to be

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 755 – 764, 2005.
© IFIP International Federation for Information Processing 2005

considered to address the issue of conserving battery power and maintaining short access latency: (i) *Access time*: Elapse time from the time a request is initiated until all data items of interest are received. (ii) *Tuning time*. Amount of time spent by the client to listen for the desired broadcast data item(s). Two modes exist for tuning time; active and doze mode. Active mode is when the client listens into the channel for the desired data item, hence costly to power consumption, while doze mode is when clients simply turned into a power saving mode. With regard to minimising power consumption, a broadcast indexing scheme will be incorporated. Indexing scheme provides accurate information for a client to tune in at the appropriate time for the required data [5]. As a result, mobile clients are able to conserve the energy of their unit by switching to power saving mode and back to active mode when the data is about to be broadcast.

In this paper, we *present* Global Indexing with data placement scheme for multi channels mobile broadcast environment, which is designed to minimise query access time, client's tuning time, and power consumption. This paper is an extension of our previous work in [13]. In the earlier work, we focus on the data items ordering and access time performance without involving any indexing scheme. We developed Global indexing scheme in [12]. However, it only concerns with the index structure without considering data instances. This paper combines both data placement and Global indexing scheme forming a single broadcast system. It should be noted that in this paper, the data and index segment are located in different broadcast channel. Figure 1 illustrates the situation when index segment and data segment are in separate channel. The Figure shows the access and tuning time to retrieve data item # 3, which starts from the time client probes into the beginning of the index channel until the desired data item has been obtained.

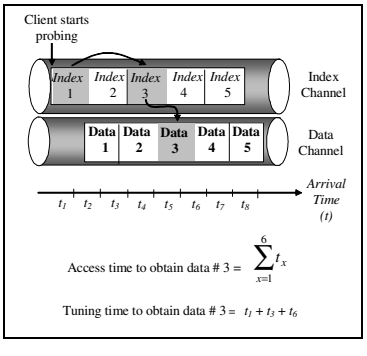


Fig. 1. Broadcast mechanism

The rests of the section in this paper are organized as follows. Section 2 contains the related work of the proposed technique. It is then followed by the description of the proposed scheme and its application in section 3. Section 4 describes the simulation model and performance results of our proposed scheme as compared to conventional method. Finally, section 5 concludes the paper.

2 Related Work

Prabhakara et al [5] presented a broadcast ordering scheme, where the hot items or the most frequently accessed data items are broadcast more often than cold items. This basic technique will be used for comparison in the later section.

Broadcast indexing technique is able to minimizing the amount of time client listening into the channel and thus less power consumption. The tree-indexing based on *B*+-tree structure is first introduced in [4]. It is then expanded and modified as what appears in [6]. This technique incorporates a separate index channel to locate the data segment in multi data channel. However, this technique does not concern with broadcast ordering issue. Our proposed technique considers broadcast indexing with data placement scheme into a single broadcast system. The proposed scheme involves single and multiple data items retrieval and allocates the data and index segment in separate broadcast channel. This fact distinguishes our paper from the existing works.

3 Global Index with Data Placement Scheme for Multi Channels Mobile Broadcast Environment: Proposed Method

3.1 Preliminaries

In this paper, we concern with request that returns multiple data items. A number of applications in this category includes a situation when mobile client wants to obtain more than one stock prices information at the same time (i.e. to list the stock price of all car companies under General Motors corp.) To accommodate this type of request, we consider the relationship between one data item and the others. Although, multiple data items retrieval is our primary objective, our proposed scheme also considers single data item retrieval. Thus, the application of this scheme is not limited to either case. Having said so, it is necessary to determine the broadcast order in advance so that the average query access time can be improved.

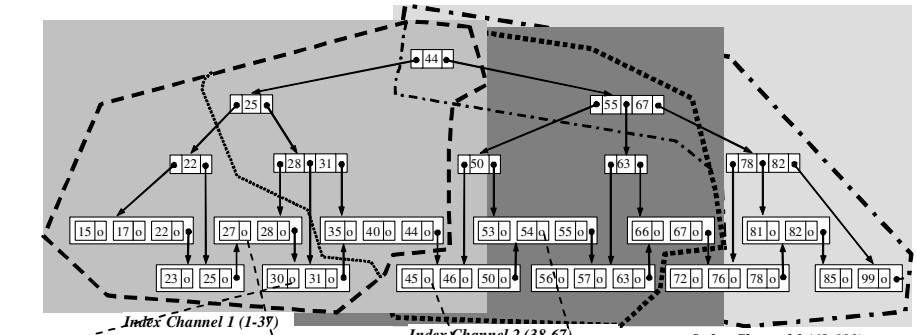
To find the query access information of mobile clients, we may incorporate either one of the following mechanism. One is to collect the access information from each mobile client at a regular interval, second is to determine the access information from the behaviour of offline or desktop and wireless users with the assumption that the same data items are accessible through the internet. We can assume that offline users and wireless users have a similar access patterns. Some analysis on query patterns and access information has been reported by Microsoft research group in [1]. Once, the access information is received by the server, the statistics will be compiled and the broadcast organisation scheme will be implemented. An example of query patterns is as follows: $Q1 = \{d_1,d_3\}$; $Q2 = \{d_1,d_3,d_4\}$; $Q3 = \{d_5\}$; $Q4 = \{d_6,d_7,d_8,d_9\}$; $Q5 = \{d_2,d_3,d_7\}$.

Having found the query access information, broadcast programs can be classified into two categories: *with replication* and *without replication*. A broadcast program with replication corresponds to the case that the data items appear with different frequencies. Generally, the most popular data item will be broadcast more often than

others. On the other hand, a broadcast program is without replication if all data items are broadcast with equal frequencies or uniform frequencies. A broadcast program without replication is also called a flat broadcast program. This paper relates to the later broadcast program and leaves the replication broadcast program for future work. Moreover, our proposed placement scheme is designed under assumption that the queries have equal access frequencies.

3.2 Global Indexing Model

Global Indexing model is designed based on *B+* tree structure [12]. Global index consists of non-leaf nodes, and leaf node. Leaf node is the bottom most index, where each key point to actual data items.



Data Channel 1			
30	Argentina	MerVal	903.680
72	Brazil	Bovespa	18,796.91
44	Canada	S&P TSX Composite	8,191.61
67	Chile	IPSA	1,412.70
53	Mexico	IPC	9,821.000
99	United States	S&P 500	1,098.70
55	Australia	All Ordinaries	3,366.800
78	China	Shanghai Composite	1,562.688
63	Hong Kong	Hang Seng	11,276.86
66	India	BSE 30	5,069.87

Data Channel 2			
25	Indonesia	Jakarta Composite	722.709
28	Japan	Nikkei 225	10,849.63
17	Malaysia	KLSE Composite	793.97
81	New Zealand	NZSE 50	2,564.730
85	Pakistan	Karachi 100	5,454.69
22	Philippines	PSE Composite	1,521.61
23	Singapore	Straits Times	1,754.96
27	South Korea	Seoul Composite	768.46
31	Sri Lanka	All Share	1,321.19
35	Thailand	SET	609.72

Data Channel 3			
46	Taiwan	Taiwan Weighted	5,777.32
50	Austria	ATX	1,798.41
54	Belgium	BEL-20	2,391.09
57	Czech Republic	PX50	750.80
76	Denmark	KFX	252.53
82	France	CAC 40	3,603.26
15	Germany	DAX	3,803.10
56	Italy	MIBTel	20,513.000
40	Netherlands	AEX General	327.39
45	Norway	OSE All Share	327.39

Fig. 2. Global Index Model

When being broadcast, each physical pointer to the neighbouring leaf node as well as actual data item are replaced by a time value, which indicates when the leaf node or data item will be broadcast. Global index scheme is partitioned into a number of channels and each channel has some degree of replication. Each index channel has a different part of the entire index structure, and the overall structure of the entire index is still preserved. As such, Global indexing scheme has the same behaviour as single channel system. Global index is exhibited in Figure 2. Further details on Global Index model can be found in [12]. Query processing in this scheme can be described as follows:

- Mobile client tunes in one of the index channel (i.e. can be of any index channel).
- Mobile client follow the index pointer to the right index key. The pointer may lead to another index channel that contains the relevant index. While waiting for the index to arrive, mobile clients can switch to doze mode.
- Mobile client tunes back on at the index segment that has the right index key, which point to the data channel that contains the desired data item. It indicates a time value of the data to arrive in the data channel.
- Mobile client tunes into the relevant data channel, and switch back to doze mode while waiting for the data item to come.
- Mobile client switches back to active mode just before the desired data item arrives, and retrieves the required items.

3.3 Data Placement Scheme

Let us denote $D = \{d_1, d_2 \ldots d_m\}$, which is a set of data items to be broadcast in the server, and Q as a set of queries $\{Q_1, Q_2, \ldots, Q_n\}$. In this case, we assume the data item has an equal size and the order of the retrieval can be arbitrary, which means if any of the required data by Q_i arrives in the channel, it will be retrieved first. Each query, Q_i, accesses a number of data items j, where $j \subset D$. The broadcast channel is indicated by C, and the length of the broadcast cycle in a channel is given by BC. We denote the broadcast schedule as $S = \{d_x, d_y, \ldots . d_z\}$. Similarly, the broadcast program for each channel is defined by SC.

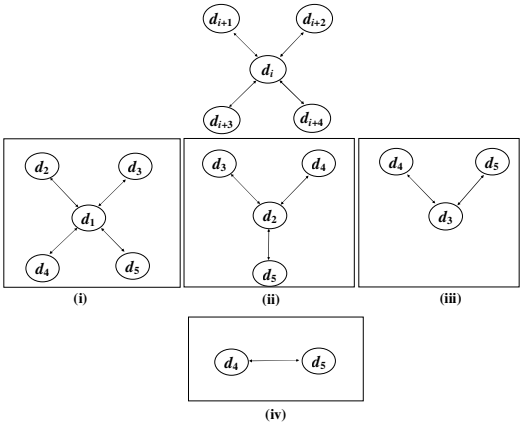


Fig. 3. The architecture of the proposed method

The mechanism of our proposed scheme is divided into three stages.

- *Firstly*, we calculate the access frequency of each data item given by Q. Subsequently, the data items will be sorted in decreasing order of value of access frequency. The result will be temporarily stored in *ST1,* and incorporated into the final broadcast order S after the second stage is finalized. The algorithm of this stage is given in Figure 4 (a).

- *Secondly*, the relationship between one data item and the other will be analyzed. The access frequency of each data item in relation with other data item will be assessed from given Q. This stage is required to allocate the related data items close to each other. Thus, the number of calculation required for this purpose will be $\frac{m \times (m-1)}{2}$, m is the total number of data items in D. Figure 4 illustrates this stage. It shows that each data item to be broadcast will be analyzed for its relationship with other data item based on Q. The calculation is done in sequence basis. i in Figure 3 reads the data item number. This process iterates until each data item has been counted. A single process includes two ways relationship of the data. Figure 3 also depicts an example when there is 5 number of data items to be broadcast.
  Each data item will be counted for its relationship with the next sequence of the data item from given Q. The double point arrows indicate two ways relationship of the two data. In this case, $D = \{d_1, d_2, d_3, d_4, d_5\}$, we count the frequency of the two data items, which appear at the same time in a query. Started from d_1, which is counted towards the relation with the next sequences of the data item that is d_2, d_3, d_4, d_5. This process continues sequentially until m-1 data item. After all data item has been analysed, they are sorted in non-increasing order based on the access frequency with other data item, and stored in *ST2*. Each allocation will involve two data items. If there is a duplication of data item within *ST2*, then data item with the highest *ST2* value are kept and the rests are removed. Figure 4 (b) shows the algorithm of the second stage.

- The *final stage* is to combine *ST1* and *ST2* forming a single broadcast order S. We start with allocating the data item in *ST2* into S also in non-increasing order. This process continues until just one access frequency left to be processed. Subsequently, we take the result from the *ST1*, analyse it against S, the data item in *ST1* that has existed in S will be discarded. The left over data item is put into S with non-increasing order as well. The broadcast order is then allocated over multiple channels. Assuming the required number of data items for each channel is known, which is indicated by *BC*. Once the *BC* is reached, the subsequent data items will be broadcast in another channel. We can employ our previous work in [11] to determine the optimum number of data items in broadcast channel. The algorithm of this mechanism is given in Figure 4 (c).

Having known the broadcast order enable us to determine the sequence of data items that minimize the average access time. The next issue to consider is to specify the broadcast program. The broadcast program indicates the schedule of the first item in S at the broadcast cycle, and the sequential items follow the order that has been generated. To achieve the most effective broadcast program, one must note the statistical patterns of users start listening or probing into the channel. In this paper, we assume the pattern follows the behavior of Gaussian distribution. We allocate the broadcast order *SC* and *SIC* into the program, where the first item in each *SC* and *SIC* is placed at point 2/3 in the broadcast program. The point is chosen to minimize the chance of clients missing the data of interest and have to wait for the subsequent

Algorithm 1. Broadcast Ordering Algorithm – Stage 1

Input: An array of data items (D), queries (Q_i) with their related data items (j_i), and the total number of data items to be broadcast (m).
Output: An array of broadcast program ($ST1$)
Procedure:
1. Initialise $i = 1$, $T = 0$
2. **Do while** i: m
3. **For** $n = 1$ to Q_{max}
4. Count the frequency of d_i in Q_n
5. Store and cumulate the result in temporary variable T
6. **If** Q_{max} then
7. $ST1_{(i)} = T$
8. Clear T
9. **End if**
10. **Next** n
11. increment i
12. **Loop**
13. Check the largest value in $ST1$
14. Ordered the value $ST1$ such that $ST1\ (1) \geq ST1\ (2) \geq \ldots \geq ST1\ (m)$
15. **End Procedure**

(a) First Stage Algorithm

Algorithm 2. Broadcast Ordering Algorithm – Stage 2

Input: An array of data items (D), queries (Q_i) with their related data items (j_i), and the total number of data items (m).
Output: Array of broadcast program ($ST2$).
Procedure:
1. Initialise i = 1, $T = 0$
2. **Do while** $i < m$
3. **For** $k = (i + 1)$ to m
4. **For** $n = 1$ to Q_{max}
5. Count the frequency of d_i retrieved at the same time
6. with d_k in Q_n
7. Store and cumulate the result in temporary variable T
8. **If** Q_{max} then
9. $TDarray\ (d_i, d_k) = T$
10. Clear T
11. **End if**
12. **Next** n
13. **Next** k
14. increment i
15. **Loop**
16. Check the largest value in $TDarray$
17. Ordered the value in $TDarray$ from the largest to the smallest value such that $TDarray\ (1) \geq TDarray\ (2) \geq \ldots \geq TDarray$ (max)
18. The coordinate value for each $TDarray$ determine the data item
19. Place the coordinate value for each $TDarray$ to $ST2$ with the non-increasing order. Check for any duplicate data item within $TDarray$. Data item can only appear one for each cycle. Thus, in case there is any duplication, data item with the highest $TDarray$ value are kept and the rests are removed.
20. **End Procedure**

(b) Second Stage Algorithm

Algorithm 3. Broadcast Ordering Algorithm – Stage 3

Input: Two arrays of Broadcast schedule ($ST1$ *and* $ST2$), optimum length of broadcast cycle in a channel (BC)
Output: An array of Final Broadcast program (S), two dimensional array of broadcast program for each channel (SC)
Procedure:
1. **For** $i = 1$ to $ST2_{(max)}$
2. Move $ST2_{(i)}$ to S
3. **Next** i
4. **For** $i = 1$ to $ST1_{(max)}$
5. Check if $ST1_{(i)}$ exists in S
6. **If** $ST1_{(i)} \subset S$ then
7. skip
8. **Else**
9. Move $ST1_{(i)}$ to S_{max+1}
10. **End if**
11. **Next** i
12. Let channel number = x
13. Initialise $x = 1$, $i = 1$, $j = 1$, $k = 1$
14. **Do While** i : $S_{(max)}$
15. **If** k : BC
16. $SC(x, j) = S(i)$
17. increment j
18. **Else**
19. **if** $k > BC$ then
20. $k = 1$; $j = 1$
21. $x = x + 1$
22. **End if**
23. **End** if
24. increment i
25. increment k
26. **Loop**
27. **End Procedure**

(c) Third Stage Algorithm

Fig. 4. Broadcast Ordering algorithms

cycle. With this way, majority of requests are served within a single broadcast cycle. Although this means about 50% of requests have to wait for a short time period but the overall performance will be better since only a fraction of requests need to wait for the next cycle. The broadcast program can then be generated at a regular interval.

4 Performance Evaluation

In this section, we analyze the performance of our proposed method in two cases. In the *first* case, we compare the proposed method against conventional ordering scheme. In this conventional scheme, the data items are ordered and broadcasted based on the access frequency without considering the relationship with other data items. The first case concerns with the access time performance. The *second* case relates to determining the tuning time of Global index and non-global index. In this last case, we also calculate the average power consumption of mobile clients for listening to the channel.

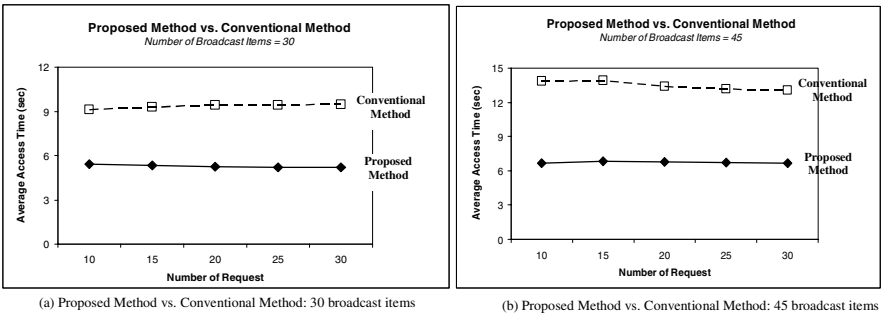
Table 1. Parameters of Concern

Parameters	Value
Size of each data Item	2KB
Bandwidth	64Kbps
Query Patterns/Profiles	10
Number of Dependent Items in Query	1-4
Number of Broadcast Channel	3
Number of Broadcast Data Items	30 and 45
Number of Request	30
Global Index	
Node Pointer Size	5 bytes
Data Pointer Size	5 bytes
Indexed Attribute Size	4 bytes
Index Arrival Rate	4 index nodes per sec

In this evaluation, we use the same data and indexing scenario as illustrated in Figure 2. The index structure is broadcasted following a breadth-first order. To analyse the schemes, we develop a simulation model. The simulation is carried out using two software packages namely Visual Basic 6.0 and *Planimate* or animated planning platforms [10]. The prototype of our system has been reported in [14]. The simulation environment is set to apply exponential distribution for data and index inter-arrival rate given an average value. In the query profile, we consider request that return one and up to four numbers of items. The parameters of concern are given in Table 1. The channel bandwidth is set to 64Kbps which follows the EDGE standard.

4.1 Access Time

Case 1: This case compares the proposed method with a conventional data ordering method as presented in [9].



(a) Proposed Method vs. Conventional Method: 30 broadcast items

(b) Proposed Method vs. Conventional Method: 45 broadcast items

Fig. 5. Proposed Method vs. Conventional Method

The conventional ordering scheme relates to the basic ordering scheme that allocates the data items in the channel according to the access frequency in a non-decreasing order and the entire index structure is broadcast within an index channel. Two analyses are also considered, which involve 30 and 45 number of broadcast items. In Figure 5, we can see clearly that the proposed method provides a better access than the conventional method with about one and half times lower access time. A similar trend occurs when the number of broadcast items increased (refer to Figure 5 (b). It is shown, that the increase of broadcast items severely affects the conventional method. In the contrary, the access time of our proposed scheme with Global Index rises just a few seconds or can be considered as minor increase. Consequently, the gap between our proposed method and conventional method is getting larger.

4.2 Tuning Time

Case 2: This case relates to comparing the tuning time of our Global index with non-global index scheme. Non-global index refers as a conventional index broadcasting method, which utilizes a single broadcast channel to broadcast the index structure. Tuning time defines the amount of time clients must listen to the channel to retrieve the desired data instances. The analysis involves 30 numbers of broadcast items. As exhibited in Figure 6 (a), the tuning time of mobile clients accessing our Global index is far less than the non-global index scheme. More importantly, the result also indicates that clients spend much less power consumptions during query operation with Global indexing scheme.

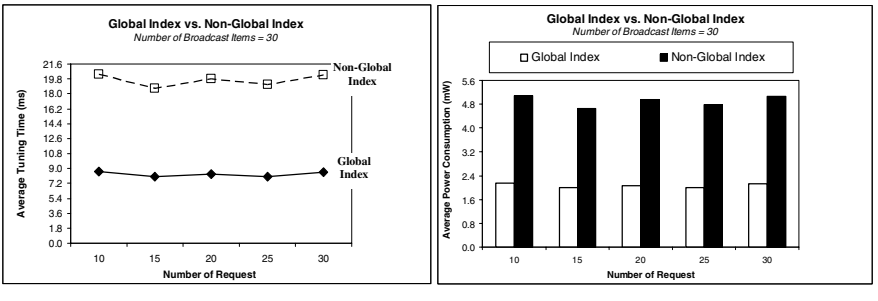


Fig. 6. Tuning time and Power Consumption: Global vs. Non-Global Index

According to Imielinski et al [3] a device with Hobbit chip (AT&T) requires about 250mW power consumption during active mode, and 50μW during power saving mode. Thus, we can see that the duration of clients' listening and not listening to the channel has affected the power consumption. We measure the power consumption of mobile clients based on the duration of active mode and power saving mode that has been obtained from the simulations. The calculation is determined from the following formula:

Power Consumption = (250 x *Time during active mode*) + (0.05 x *Time during power saving mode).* As depicted in Figure 6 (b), the power consumption of clients

accessing the Global index is substantially less than non-global index. With little power consumptions, client can reserve battery power more efficiently, which is very much desirable considering the lack of resources in mobile devices.

5 Conclusions and Future Work

In this paper, we have presented Global indexing with data placement scheme for multi broadcast channel environment. We have developed a simulation model to analyze the performance of our proposed scheme. We compare the proposed scheme with conventional scheme. The analysis includes access time, tuning time and power consumption. It is found that the proposed scheme provides substantially better performances in every aspect of evaluation.

For future work, we will incorporate a certain degree of data replication in the broadcast program. We will investigate the most effective degree of the replication as well as the structure and scheduling of the broadcast program.

References

1. Adya A., Bahl P., Qiu L., “Analyzing the Browse Patterns of Mobile Clients” In *Proc. of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pp.189-194, 2001.
2. Barbara D., "Mobile Computing and Databases-A Survey", *IEEE TKDE*, **11**(1):108-117, 1999.
3. Imielinski T., Viswanathan S. and Badrinath B. R., "Data on Air: Organisation and Access", *IEEE TKDE,* **9**(3): 353-371, 1997.
4. Imielinski T., Viswanathan S. and Badrinath B. R., "Energy Efficient Indexing on Air", In *Proc. of the ACM Sigmod Conference***,** pp.25-36, 1994.
5. Lee K. C. K., Leong H. V. and Si A., “Semantic Data Access in an Asymmetric Mobile Environment”, In *Proc .of the 3rd Mobile Data Management***,** pp.94-101, 2002.
6. Leong H. V. and Si A., "Data Broadcasting Strategies Over Multiple Unreliable Wireless Channels", In *Proc. of the 4th CIKM***,** pp.96-104, 1995.
7. Myers B.A. and Beigl M., “Handheld Computing”, *IEEE Computer Magazine*, **36**(9):27-29, 2003.
8. Paulson L.D.,”Will Fuel Cells Replace Batteries in Mobile Devices?” *IEEE Computer Magazine*, **36**(11):10-12, 2003.
9. Prabhakara K., Hua K.A, and Jiang N., “Multi-Level Multi-Channel Air Cache Designs for Broadcasting in a Mobile Environment”, In *Proc. of the ICDE*, pp. 167-176, 2000.
10. Seeley D. et al, *Planimate^{tm}-Animated Planning Platforms*, InterDynamics Pty Ltd, 1997.
11. Waluyo, A.B., Srinivasan B., Taniar D., “Optimal Broadcast Channel for Data Dissemination in Mobile Database Environment”, In *Proc. of the 5th APPT*, pp. 655-664, 2003a.
12. Waluyo, A.B., Srinivasan B., Taniar D., “Global Index for Multi Channels Data Dissemination in Mobile Databases”, In *Proc.of the 18th ISCIS,* pp.210-217, 2003b.
13. Waluyo, A.B., Srinivasan, B., and Taniar, D., "Allocation of Data Items for Multi Channel Data Broadcasting in a Mobile Computing Environment ", In *Proc. of the 1st EUC*, pp. 409 – 418, 2004.
14. Waluyo, A.B., Goh, G., Srinivasan, B., and Taniar, D., “On-Building Data Broadcast System in a Wireless Environment, *International Journal of Business Data Communications and Networking*, **1**(4), 14-36, 2005.

An Adaptive Mobile Application Development Framework

Ming-Chun Cheng and Shyan-Ming Yuan

Department of Computer and Information Science, National Chiao Tung University,
1001 Ta Hsueh Rd, Hsinchu 300, Taiwan
{native, smyuan}@cis.nctu.edu.tw

Abstract. Although wireless networks and mobile devices become popular these days, the diversity of mobile devices and unsteadiness of wireless networks still cause software development much trouble. Thus, when developing a mobile application, developers are forced to expose to these problems and to be familiar with these technologies and therefore it will spend a lot of time to write a mobile application. Furthermore, a mobile application often needs to be ported to different platform (for example from Java to .NET) which is also burdensome. In order to overcome these problems, the author proposes an adaptive framework to help developers build mobile application effortlessly and rapidly. The mobile application developed on this framework can run in different devices and operating system, so developers does not need to worry about portability. As a result, a mobile application developed on this framework can enjoy the benefit of "write once, run everywhere, and access any services."

1 Introduction

In the past decade, the computing environment has changed tremendously. The bandwidth of wired network increases from 100M to Giga bps, and wireless networks become more and more popular. People often talk about mobile or pervasive computing, because mobile handheld devices get much more powerful and cheaper.

Accordingly, much ink has been spent on pervasive computing. In fact, it is not really so close to us. Owing to the size and battery consumption, processing powers of handheld devices are still insufficient to replace desktop computers do. In addition, the diversity of the computing environments is also a big trouble. Developers have to consider the differences among them and portability is an important issue. Thus, there are still considerable dilemmas needed to be overcome.

In order to solve the problems, the concept of virtual home environment (VHE) has been proposed by 3GPP. Although a large number of studies have been made on how to achieve VHE, such as MExE, OSA and USAT, little is known about providing an adaptive mobile application development framework by leveraging thin-client computing concept [1][2][3]. X window [17][20] and web technologies [4][5][6][7][8][9] are classic samples of thin-client computing. Nevertheless, the previous lacks support for handheld devices and the later lacks better interactions with end-users.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 765 – 774, 2005.
© IFIP International Federation for Information Processing 2005

The aim of this paper is to design and implement a mobile application development framework capable of supporting heterogeneous environments, and the system is called ART. The main idea is to divide an application into two parts, UI and logic, in development time and run time. This idea makes development easier and adaptation possible. Hence, in this framework, the same application could be written once and used everywhere and developers have not to consider portability.

The rest of this paper is organized as following. In the next chapter, system architecture is introduced. Next, adaptation mechanisms are presented in chapter 3 and the workflow of the development process is illustrated in chapter 4. Finally, chapter 5 shows conclusions and future works.

2 System Architecture

First of all, we have to explain the scenario how end-users put our system to use. End-users might own not only desktop computers but also several mobile devices, such as WAP phone, Java-enabled handset, Smart phone, PDA and so on. When outdoors, they probably need some resources or continuing some programs on their desktops and might wish to control appliances, i.e. TV, air conditioner and light, through their mobile devices also. According to the previous scenario, the authors leverage a 3-tiers architecture to fit these requirements. It includes front-tier, middle-tier and backend-tier; it is represented diagrammatically in Fig.1.

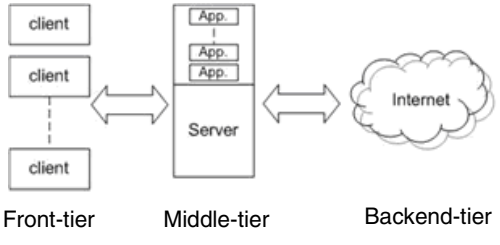


Fig. 1. The 3-tiers architecture

In this architecture, end-users use their own mobile devices or desktops in front-tier to access resources provided by applications run in middle-tier. In addition, the applications might access third-party services, e.g. X10, UPnP and web services [21], in backend-tier by calling some system services provided by server.

According the 3-tier architecture, ART system is designed as a client-server model, shown in Fig.2, and participates in the two tiers of the architecture, front-tier and middle-tier. Client run in front-tier is responsible for user interface as well as server run in middle-tier is responsible for logic computing. They communicate with each other by using asynchronous message delivery mechanism (the top part of the Fig.2) or other existent protocols (the bottom part of the Fig. 2), for example HTTP. In other words, this model could be viewed as a thin-client computing with adaptive capability. This

section will present individual parts first and then introduce how they cooperate with each others. In order to simplify terminologies, in ART system, the server is called ARTServer, the client is called ARTClient, and the application is called ARTApp.

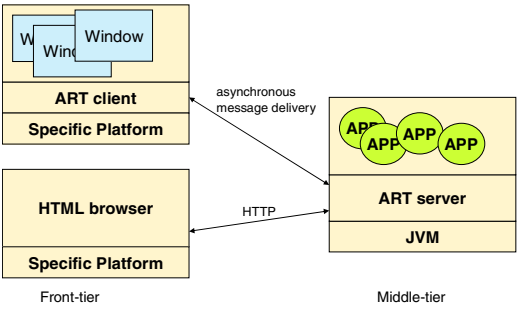


Fig. 2. The client-server model in ART system

2.1 Asynchronous Message Delivery Mechanism

Since wireless networks are not stable enough, the authors designed an asynchronous message delivery mechanism for transmission between ARTServer and ARTClient. This mechanism decouples the ARTApp and low-level network protocols, so it can help ARTServer to handle the situation of disconnection and prevents ARTApp from accessing network directly. Besides, this mechanism provides the communication capability among ARTApps also and it supports one-to-one, one-to-many and many-to-many modes. The four main components in this mechanism are ARTMesg, Queue, Message Dispatcher and Daemon Thread. ARTMesg is a message which carries some information about source, destination, command, request and user etc. For example, when user presses a button in client-side or ARTApp orders the client-side to create a new UI widget, the corresponding ARTMesg will be generated and then routed to correct place. Moreover, Queue is a data structure capable of storing and aggregating the ARTMesg. Finally, Message Dispatcher and Daemon Thread are both working threads which route or forward the ARTMesgs put in the Queue. The details will be described in section 2.4.

2.2 ARTApp

The applications which comply with ART Programming Framework run on ARTServer are called ARTApp and they might provide different services for ARTClient, e.g. search engine, appliance controller, or games. In development time, an ARTApp is separated into two parts, UI and logic, to cut down the development time. ART system leverages XUL to describe the user interface and exploits Java Language to write the logic part. The details about how to write an ARTApp is presented in Chapter 4.

In Runtime, every ARTApp owns an independent runtime space but they could communicate with others through asynchronous message delivery mechanism

mentioned previous. Furthermore, an ARTApp owns a Windows Manager (WinMngr) to maintain the relationship with its UI widgets described in XUL files.

In ART system, each ARTApp is encapsulated as a Java Jar file. When necessary, it is loaded dynamically by ARTServer and ARTServer will assign a thread to run it.

2.3 Backend-Tier

There are many resources in internet or other networks, such as X10, UPnP and Jini. In ART system architecture, these resources are all in backend-tier. Currently, ART system only provides different APIs to access these various resources.

2.4 Middle-Tier (ARTServer)

The ART system in this tier includes two components, ARTServer and ARTApp. ARTServer provides the runtime environment for ARTApp and ARTApp can use these services provided by ARTServer to access what they want. The architecture of ARTServer is showed in Fig. 3 and it could be considered as a layered architecture; they are Application Runtime Layer, Message Routing Layer and Adaptive Transport Layer. The authors hope ART system can be modularized very well and the layered design makes maintenance and upgrade easier.

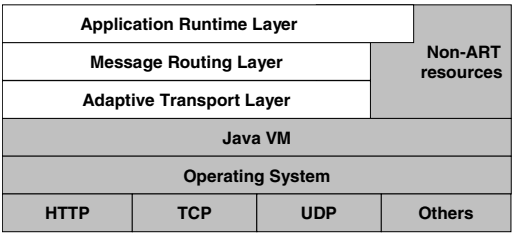


Fig. 3. The layered architecture of ARTServer

2.4.1 Adaptive Transport Layer

Adaptive Transport Layer gives ARTServer the ability to communicating with different kinds of ARTClients. Fig. 4 may help us to understand the detailed structure of ARTServer.

The main role of this layer is Communication Manager (CommMngr) which is a super daemon capable of handling many different protocols, such as TCP, UDP, HTTP and cHTTP, and it has two missions. First, it establishes the relationship between ARTServer and ARTClient when ARTClient sent the login request to ARTServer. Secondly, if login successful, CommMngr creates a logic process (i.e. a user process, including a UserOutD, a UserInD and a UserOutQ) for that ARTClient. Every logic process might have different components or functionalities depending on what kind of ARTClient it serves. UserOutD is responsible for picking ARTMesgs from UserOutQ, and sending

them to the corresponding ARTClient or translating ARTMesgs to specific format. The UserInD handles or translates incoming requests from its client and put them into InnerQueue. The details of adaptation mechanisms in this layer will be introduced in chapter 3.

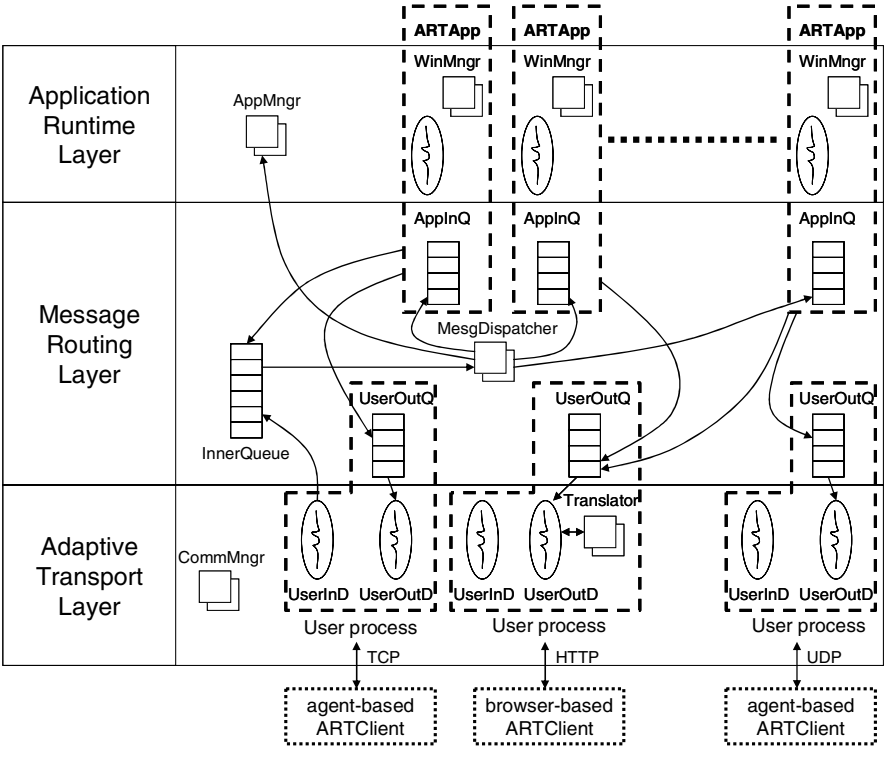


Fig. 4. The detailed architecture of ARTServer

2.4.2 Message Routing Layer

This layer is the core of asynchronous message delivery mechanism. The main components are Queue and Message Dispatcher (MesgDispatcher), which is responsible for routing ARTMesg to the correct queue. There are three kinds of queues on ARTServer: InnerQueue, AppInQ and UserOutQ.

All the messages received by UserInD are placed in InnerQueue. Then MesgDispatcher will dispatch them to the some AppInQ which ARTApp will handle these ARTMesg in.

Once an ARTApp generates an ARTMesg whose destination is an ARTClient, the message will be placed in UserOutQ of the user process that serves the client. UserOutD in this user process will send the message to its client later.

However, an ARTApp might need to negotiate with other ones on the same ARTServer; ARTMesgs in this situation will be put back to InnerQueue and wait for dispatching by MesgDispatcher again.

2.4.3 Application Runtime Layer

An ARTServer can serve many different ARTClients at the same time, and each ARTClient can also have many ARTApps run on the server at the same time. Application Manager (AppMngr) is responsible for loading, resuming and stopping ARTApps. Before starting an ARTApp, AppMngr will check if any instance of the ARTApp already exists in the memory. If it does, AppMngr will then check what running mode of the ARTApp is and decide to create a new instance or bind the client to the existent one.

2.5 Front-Tier (ARTClient)

The front-tier part of ART system is called ARTClient which is responsible for presentation. It provides user interface to receive orders from end-users and displays results from middle-tier. In ART system, end-users can use many kinds of devices in front-tier and ART system could automatically tailor results to fit diverse environments. In these days, ART system can support four popular kinds of devices, including J2ME MIDP [12][13][14], .NET CF, WAP and HTML browser. Because of diversity of these devices, authors divide these devices into two categories, agent-based and browser-based, and use different mechanisms to design and implement them.

An ARTClient can access many ARTApp simultaneously, so ART system provides a facility, shown in Fig. 5, like Task Manager within Microsoft Windows XP. It helps end-users to select what ARTApp to start, stop or switch to.

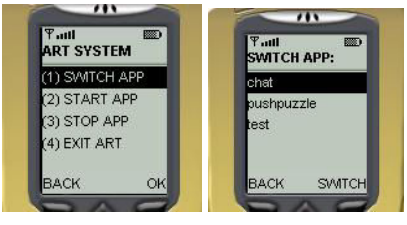


Fig. 5. The screenshot of the task manager within ART system

2.5.1 Agent-Based Client

An agent-based client is defined as a device capable of executing author-developed ARTClient on it. According the definition, it has to provide some programming environments for programmers to develop some programs; nowadays, the most popular programming environments of handheld devices are J2ME MIDP and .NET CF. In order to support both them, two different agent-based clients are implemented and they are all compliant with "ART agent-based client specification".

2.5.2 Browser-Based Client

A browser-based client is defined as a device which is browser enabled. In other words, end-user could use these existent browsers to access many sorts of services in internet. Currently, almost mobile devices are browser enabled. Moreover, owing to adaptive transport layer, ART system can be accessed by these clients.

3 Adaptation Mechanisms

In ART system, adaptation is applied in two positions, network protocols and UI, and both two are implemented in Adaptive Transport Layer. This chapter will introduce how adaptation mechanisms work.

3.1 Network Protocols

Because of diversity of mobile devices, the networks capabilities are not the same. Some devices support TCP connection but some devices might support only HTTP. In order to hide the detailed from mobile application developers, the ARTServer exploits a loosely couple design and the adaptive transport layer is proposed. The layer defines the unified transport interface and the implementation which complies with it could be plugged into ART system easily. At the moment, ART system supports TCP, UDP and HTTP.

3.2 User Interface

In ART system, ARTApp doesn't know what kind of client it servers and it only handles and generates ARTMesgs no matter the type of client. Hence, in order to serve browser-based clients, ART system needs a mechanism to translate ARTMesgs to other formats, such as WML or HTML, and this mechanism is implemented in adaptive transport layer also.

In the section 2.4.1, the CommMngr is introduced and one mission of it is to create a user process for a client. When a client sends login request to ARTServer, CommMngr will analyze CC/PP [11] profile provided by the client. If the client is browser-based, a translator component is within the user process and it is combined with many convert functions and a layout manager.

Every convert function has different capability to translate a XUL [10] widget to the corresponding widget of other formats. Table 1 is a widget mapping among XUL, MIDP and WML.

Table 1. The mapping table among XUL, MIDP and WML

XUL	MIDP	WML
<window>	Form	<wml>+<card>
<canvas>	Canvas	<wml>+<card>+4 direction button+<img src=””>
<listbox>	ChoiceGroup	<select>
<button>	Buttom	<a href>
<textbox>	TextField	<input>

The layout manager is similar to WinMngr within agent-based ARTClient but it is capable of arranging the positions of the widgets. Another functionality of layout manager is to call convert functions to translate XUL widgets to specific formats. When serving a browser-based client, ARTMesg does not send to client directly instead of being handled by the layout manager.

4 Programming Framework

This chapter shows how to write an ARTApp. An ARTApp is combined with two different parts, UI and logic, and these two parts could be developed simultaneously by different developers. The development workflow is showed in Fig. 6.

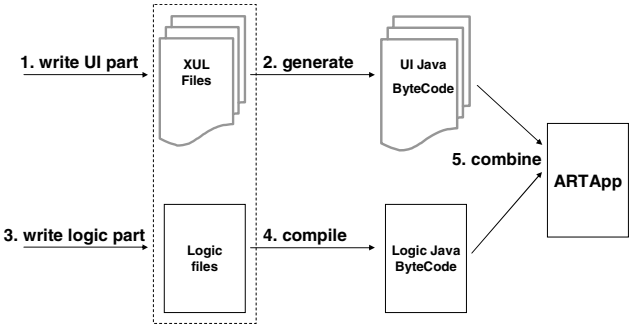


Fig. 6. The application developers have to writes two kinds of files within the dashed box

4.1 Write User Interface Part

In ART system, developers have to use XUL to describe the user interface. Fig. 7 shows a sample; left is a XUL file and right is the presentation result.

When finishing the XUL files, developers use code generator provided by ART system to generate some Java Bytecode and these codes can be used by logic part.

```
<?xml version="1.0"?>
<window id="Example">
        <listbox id="CHOICE" label="Choose one">
                <listitem id="1" label="1"/>
                <listitem id="2" label="2"/>
        </listbox>
      <textbox id="INPUT" label="Input Here"/>
      <button id="BUTTON" label="OK"/>
</window>
```

Fig. 7. A sample XUL file and the presentation result

4.2 Write Logic Part

In ART system, Java language is used to write logic part and developers who are familiar with Java have not to learn anything. The only requirement is that the main class of application has to extend the ARTApp class. Fig. 8 shows an example.

```
public class Main extends ARTApp{
    Example window=null;
    public void startApp(){
        //get the reference of the window named "Example"
        window=Example.getInstance();

        //create a listener
        class Listener implements ArtButtonListener{
            public Listener(){}
            public void actionPerform(){
                //change the text of the button named "BUTTON"
                window.BUTTON.setText("Hello");
            }
        }

        //register a listener to the button named "BUTTON"
        window.BUTTON.setArtButtonListener(new Listener());
        //bring the window named "Example" to foreground
        window.show();
    }
    public void stopApp(){}
}
```

Fig. 8. A logic codes written by Java language

5 Conclusions and Future Works

So far, we have described our system, ART system, and how it achieves the objectives mentioned in chapter 1. The key point of design is separating GUI from the application body. Developers write UI descriptions in XUL instead of writing them in program bodies. Then ART will generate the UI objects automatically according to XUL documents. Users can launch numerous applications via a small-size client on mobile devices and ART system will automatically tailor the result to fit different clients.

In the future, the authors hope the agent-based clients could be more powerful. The idea is that ART system can send some scripts to ARTClient and some logics could be run in client-side. This method makes the ART system more flexible and more efficient.

Another issue is how to integrate the services in backend-tier. Currently, the authors are designing a unified interface capable of accessing Jini, UPnP and X10 by a unified method.

References

1. Krikke, J., “Thin clients get second chance in emerging markets”, Pervasive Computing, IEEE, Volume 3, Issue 4, Oct-Dec 2004, pp. 6-10
2. Richardson, T., Stafford-Fraser, Q., Wood, K.R., Hopper, A., “Virtual network computing”, Internet Computing, IEEE, Volume 2, Issue 1, Jan.-Feb. 1998, pp. 33-38

3. Volchkov, A., “Server-based computing opportunities”, IT Professional, Volume 1, Issue 2, March-April 2002, pp. 18-23
4. L. C. Martins, “A Framework for Filtering and Packaging Hypermedia Documents”, Adaptive Hypermedia and Adaptive Web-Based Systems Second International Conference, Malaga Spin, May 2002, pp. 274-283
5. Ricky Robinson., “Context Management in Mobile Environments”, World Wide Web, http://www.itee.uq.edu.au/~ricky/thesis.doc, October 2000.
6. Mark H. Bulter., “Current Technologies for Device Independence”, World Wide Web, http:// www-uk.hpl.hp.com/people/marbut/currTechDevInd.htm, March 2001.
7. Mark Bulter, Fabio giannetti, roger gimson, and Tony wiley, “Device Independence and the Web. IEEE internet computing”, October 2002.
8. W3C, “Device independence working group charter”, World Wide Web, http://www.w3.org/2002/06 /w3c-di-wg-charter-20020612.html .
9. E. Kirda, C. Kerer, “Supporting multi-device enabled Web services: challenges and open problems”, 10th IEEE Intl. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Cambridge MA, USA, June 2001, pp. 49-54
10. eXtensible User Interface Language, http://www.xulplanet.com/tutorials/xultu/.
11. W3C, “Composite Capability/Preference Profiles (CC/PP): A user side framework for content negotiation”, http://www.w3.org/TR/NOTE-CCPP/.
12. Sun Microsystems, “Java 2 Platform Technologies”, http://java.sun.com/.
13. Sun Microsystems, “Java 2 Platform, Micro Edition (J2ME)”, http://java.sun.com/j2me/.
14. Sun Microsystems, “Mobile Information Device Profile (MIDP)”, http://java.sun.com/products/cldc/.
15. Sun Microsystems, Java Technology and Web Services, http://java.sun.com/webservices/.
16. 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Virtual Home Environment / Open Service Architecture (3G TS 23.127 version 1.10.0).
17. X.Org, “X11 technologies”, http://www.x.org/X11.html
18. P. de Bra, et al., Adaptive Hypermedia and Adaptive Web-Based Systems, Springer, May 2002.
19. User Interface Markup Language (UIML), http://www.uiml.org/index.php.
20. R. W. Scheifler, “X Window System Protocol, X Consortium Standard, X Version 11, Release 6”, Massachusetts Institute of Technology, Laboratory for Computer Science, 1994
21. W3C, Web Service Architecture, http://www.w3.org/TR/2003/WD-ws-arch-20030514/.

Context-Aware Emergency Remedy System Based on Pervasive Computing

Hsu-Yang Kung, Mei-Hsien Lin, Chi-Yu Hsu, and Chia-Ni Liu

Department of Management Information Systems,
National Pingtung University of Science and Technology,
1 Hsueh Fu Rd., Nei Pu Hsiang , Pingtung, Taiwan, 91207
{kung, mslin, m9356020, m9156014}@mail.npust.edu.tw

Abstract. This study proposed a Context-Aware emergeNcy rEmedy system (CANE), based on the operations of a real hospital, to provide complete and convenient functions for emergency processes and medical consultants using mobile communication networks. The CANE system combines emergency medical services with GIS/GPS technology, mobile multimedia communications and context-aware controls. The CANE system largely focuses on allowing the EMT personnel to arrive at the accident location in the shortest possible time and give first aid the proposed dispatching procedure and the GPS/GIS location-aware service. When an ambulance is on the way to the specified hospital, the EMT can deliver the patient's symptoms, including the data of the medical equipment and the audios/videos of the patient, to the hospital doctor via the personal communication network, e.g., GPRS. Based on real-time multimedia information, a physician can recommend to EMT the most suitable treatment for the patient on the way to the hospital.

1 Introduction

With the wide deployment of wireless/mobile communications and the increase in the number of wireless/mobile users, mobile information services, such as the telemedicine application system over mobile medical network [1], [2], [3], [4], [5], [6], have become popular. For mobile telemedicine services [7], [8], [9], users can apply handheld devices, e.g., Smart Phone, PDA, or Tablet PC, to connect with the medical network and to access medical information via the on-line hospital. These pervasive healthcare and pervasive telemedicine services can improve communications among patients, physicians and other healthcare workers, and enable accurate medical information to be delivered anytime and anywhere, thereby reducing errors and improving access [10], [11], [12].

As human society progresses, levels of natural and man-made calamities/accidents increase significantly [13]. In a natural or man-made accident, the arrival of the Emergency Medical Technician (EMT) arrives at the accident location quickly to treats the patient is the most important for saving a patient's life. However, some situations are unpredictable and uncontrolled [14], [15], [16]. For example, the ambulance driver might not know the correct accident location very well, or the emergency center might dispatch several ambulances for the same emergency [17],

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 775 – 784, 2005.
© IFIP International Federation for Information Processing 2005

[18]. Such uncontrollable situations prevent saving the patient in time and waste the social and medical resources [12], [19]. In contrast to a conventional emergency rescue where the EMT person lacks expertise and is unfamiliar with the patient's health history, the EMT person can call doctors at the hospital to obtain treatment suggestions [20]. However, the information from the distance ambulance is insufficient to help doctors make an accurate diagnosis and recommend the appropriate treatment. This uncertainty and lack of instant diagnosis can seriously jeopardize the patient's life [3], [21], [22]

This study proposes a Context-Aware emergeNcy rEmedy system (CANE) to achieve efficient ambulance dispatching and real-time multimedia symptom delivery for emergency rescues. CANE is also a mobile telemedicine system supplying a platform by which the hospital physicians can communicate with the EMT person on the ambulance and give appropriate treatment suggestions to save the patient on the way to the hospital. CANE uses GIS/GPS technologies to achieve the location-base services, which could help the rescue center to dispatch ambulances appropriately for all emergencies [23].

The rest of this study is structured as follows. Section 2 describes the infrastructure and system components of the proposed CANE system. Section 3 describes implementation and the usage of the CANE system. Conclusions are finally drawn in Section 4.

2 System Architecture Design

This section describes the system architecture and components of the CANE system, and the procedure of dispatching ambulances among the patients, the hospital and the rescue center.

2.1 CANE System Architecture

CANE is a three-tier architecture composed of (i) the Front Emergency Point (FEP), (ii) the Emergency Control Center (ECC) and (iii) the Emergency Medical Service Hospital (EMSH). Figure 1 depicts the proposed system architecture. Each subsystem in CANE is described as follows.

1. Front Emergency Point (FEP) subsystem: The FEP subsystem in each ambulance helps the EMT staff to complete emergency operations. The GIS/GPS function of the FEP subsystem helps the ambulance driver find the exact location of the patient and hospital in time. Based on the FEP subsystem, the EMT can send the real-time data about the symptoms, pictures and videos and audios of the patient to the hospital. The doctor at the hospital diagnoses the patient from a distance and gives appropriate treatment suggestions to the EMT staff. To promote the mobility capability of the ambulance and emergency person, Tablet PC and PDA combining mobile communication network, e.g., GPRS, are applied in the FEP subsystem. Because the GPRS bandwidth is limited and critical, the related GIS information is built into the FEP database to reduce the bandwidth consumed.

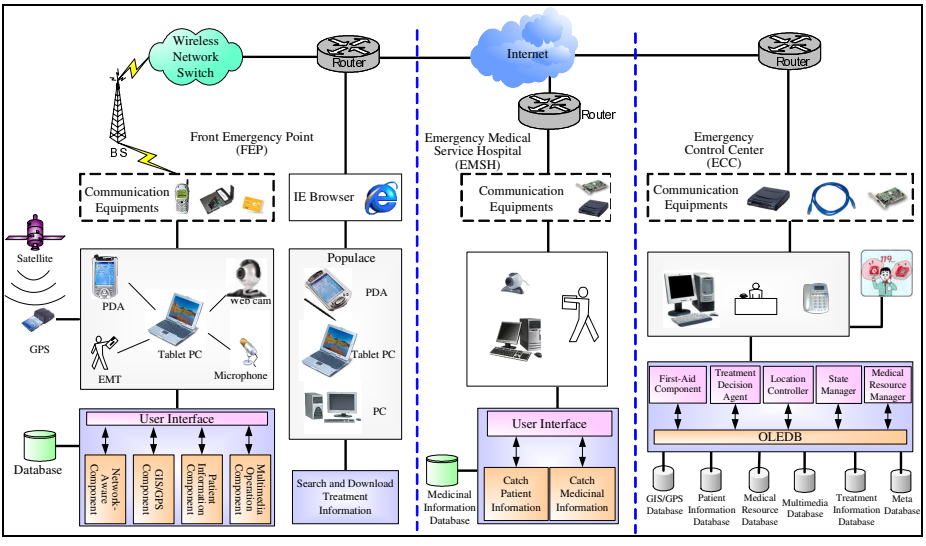


Fig. 1. The CANE system architecture

2. Emergency Control Center (ECC) subsystem: The ECC subsystem coordinates the communications between the FEP and EMSH subsystems. The ECC subsystem constantly collects the information of hospitals' medical resources and ambulances' locations. The context-aware capability of the ECC subsystem can efficiently dispatch ambulances to victims and then to the appropriate hospital, which provides sufficient medical resources and spaces. Based on the proposed dispatching procedure, the ECC subsystem filters the calls for the same incident to prevent multiple ambulances from being dispatched to the same emergency.
3. Emergency Medical Service Hospital (EMSH) subsystem: The EMSH subsystem established in each hospital constantly receives the information of patients' conditions and ambulances' locations. The distance information gives the hospital a head start in assessing the patient and preparing appropriate treatment to ensure successful emergency rescue.

CANE system combines the wire and mobile communication networks to let emergency staff and doctors communicate with each other and help treat the patient. Figure 2 shows the CANE system dispatching process. The operation steps are described as follows.

1. Step 1: ECC subsystem constantly collects the resources of hospitals and the locations/states of ambulances using the messages Inf_A(state, location) and Inf_H(resource).
2. Step 2: When ECC subsystem receives an emergency call and the patient needs an ambulance to carry him to a hospital, the ECC subsystem determines an appropriate ambulance and tells the ambulance to go to the patient's location.
3. Step3: When the EMT staff member in the ambulance receives the message Msg_A(patient information) from the FEP subsystem, he responds by accepting or rejecting the rescue mission to the ECC subsystem via the message ACK_ A (accept, state).

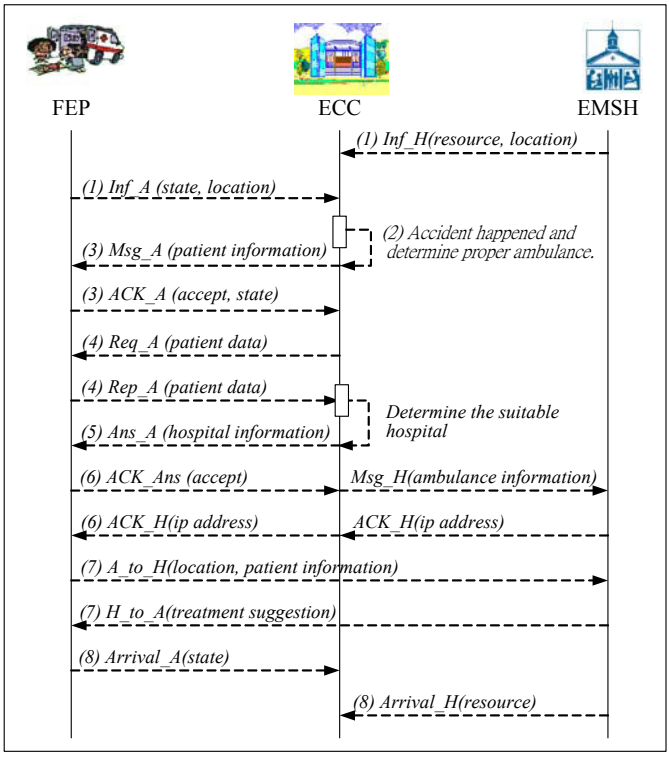


Fig. 2. CANE dispatching process

4. Step 4: The ECC subsystem asks the FEP subsystem to transmit the information of the patient. Therefore, when the ambulance arrives at the accident location, the EMT person transmits the symptoms, including the patient's pictures/videos, from the web cam on a handheld device, to the ECC subsystem.
5. Step 5: The ECC subsystem determines the appropriate hospital according to information received from the FEP subsystem, and directs the ambulance driver.
6. Step 6: When ambulance drivers request to go to the emergency hospital, the ECC subsystem informs the selected hospital and gives the doctors information of the patient. Then, the ECC subsystem establishes a communication channel between the FEP subsystem and the EMSH subsystem, and the EMT staff and doctors can then talk with each other.
7. Step 7: When the communication channel is established between the FEP subsystem and EMSH subsystem, an EMT staff member transmits the patient's symptoms, including pictures, videos and EKG, to doctors at the hospital. Physicians can then immediately diagnose and give appropriate treatment suggestions. Thus, the hospital staff can assess the patient and prepare treatment before the patient arrives.
8. Step 8: When the ambulance arrives at the selected hospital (EMSH subsystem), the FEP subsystem transmits the message Arrival_A(state) to announce that the

first-aid mission is accomplished. The ambulance is thus available to provide assistance on another emergency. The EMSH subsystem also transmits the message Arrival_H(resource) to update the medical resource database in the ECC subsystem. After the appropriate calculation and update, the ECC subsystem can determine an appropriate EMSH subsystem for the next emergency rescue.

2.2 Functionality of CANE System

This sub-section describes the designed functions of the CANE system in detail.

1. Front Emergency Point (FEP) subsystem: FEP provides four main functions, (i) location-aware operations for ambulances, (ii) patient information delivery, (iii) multimedia symptom capability and (iv) network-aware operations. The corresponding system components achieving the four functions are described as follows.
   - The GIS/GPS component provides the location-aware operations for ambulances. This GIS/GPS component obtains the location information from the embedded GPS device. The real-time location information helps the ambulance driver to find the right route quickly.
   - The Patients Information component records the patient information. When the ambulance is on its way to the emergency hospital, the EMT person records the patient information, e.g., personnel information, EKG and blood pressure, and then transmits the collected information to physicians via the mobile network. The information enables hospital staff to start assessing the patient and preparing treatment.
   - The Multimedia Operation component enables symptoms to be depicted using multimedia. The text/voice-based information is usually insufficient to help doctors to perform correct diagnosis and treatment suggestions from a distance. Multimedia symptom information, including pictures and videos of the patient's symptoms, provides doctors with comprehensive information, allowing them to make correct and quick diagnosis and treatment suggestions for the EMT person, and the more preparing time for the doctors themselves, and may make the difference between life and death.
   - The Network-Aware component detects the network connection status. This component ensures stable communication, especially given the large volume requirement of multimedia information and the limited mobile network bandwidth.
2. Emergency Control Center (ECC) subsystem: The ECC subsystem is composed of (i) the First-Aid Information Component, (ii) the Treatment Decision Agent, (iii) the Location Controller, (iv) the Medical Resource Manager and (v) the State Manager.
   - First-Aid Information Component: If the FEP person requires the first-aid information, the FEP person transmits the patient's symptoms to the ECC subsystem. The First-Aid Information Component responds with the first-aid operations to the EMT person. The multimedia database supplies the multimedia

first-aid information. The FEP person can easily understand the operation first-aid steps from the Virtual Reality (VR) demonstrations or pictures.

- Treatment Decision Agent: Treatment Decision Agent provides the EMT people and doctors with the required treatment information. The EMT person and doctor can quickly obtain the patient's anamnesis from the Treatment Decision Agent, which uses the patient information database to record the patients' health histories, e.g., name, sex and allergies. Treatment Decision Agent also provides physicians with appropriate treatment suggestions, including the appropriate emergency hospital (EMSH subsystem), based on the treatment information and medical resource databases.
- Location Controller: The ECC staff dispatch ambulances by monitoring the ambulances' locations and the corresponding emergency states. The Location Controller determines the closest ambulance for an accident, and provides the ambulance driver with appropriate routes to the dedicated hospital based on GIS/GPS database. The GIS/GPS database records the GIS map and the paths of the hospitals locations to accomplish a quick rescue.
- Medical Resource Manager: The Medical Resource Manager collects the medical resources of EMSHs, e.g., the number of available sickbeds, the medical specialty and the hospital location based on the medical resource database records.
- State Manager: State Manager processes the states of an emergency mission, which include the processing authority of the FEP system between the ECC subsystem and the EMSH subsystem. The related state information is recorded in the meta-database.

3. Emergency Medical Service Hospital (EMSH) subsystem: The EMSH subsystem established in each hospital constantly receives information about patients' conditions and ambulances' locations. The EMSH subsystem receives the pictures and real-time A/V streams about the patient from the ambulance. Then, the EMSH specialists can diagnose and give treatment suggestions to improve the patient's chance of survival. The EMSH subsystem estimates the arrival time of the ambulance, allowing the hospital staff could prepare treatment in advance for the patient. The EMSH subsystem also automatically transmits the hospital's medical resource status to the ECC subsystem for medical resource management.

3 The CANE System Implementation and Usage

The feasibility and practicality of the CANE system was considered by discussing its design and development with the staff of a teaching hospital. CANE system is based on the operations of a real hospital, providing complete and convenient functions for emergency processes and medical consultants via mobile communication networks. The CANE software and hardware configuration is described as follows.

1. System software: CANE was developed using Visual C++, Visual studio .NET 2003, eMbedded Visual Basic, ArcGIS 8.0 and MS SQL 2002.
2. System hardware: The equipment applied included Tablet PC, PDA and Web Cam for CF/USB, wireless microphone/earphones, and GPRS module.

Figures 3, 4 and 5 depict the CANE system usage. Figure 3 depicts the ambulance EMT using a PDA to record the patient's personal information, such as sex, age and symptoms. The EMT immediately transmits the patient's medical records to the hospital to maximize the preparation time available to treat the patient. Furthermore, if the provision of medical records is insufficient to help hospital physicians diagnose the patient, then the CANE system provides the picture and whiteboard functions for the EMT personnel to show the patient and describe the patient's photograph using the whiteboard. According to Fig. 4, a physician can easily understand and observe that the patient's left hand was fractured and, then, give the appropriate treatment suggestions to the EMT personnel.

Figure 5 depicts the GIS/GPS location-aware service for the ambulance driver. Using the location-aware service, the ambulance driver knows the exact locations of the accident and hospitals, driving direction, expected arrival time, driving speed,

Fig. 3. A patient's medical records illustrated in a PDA

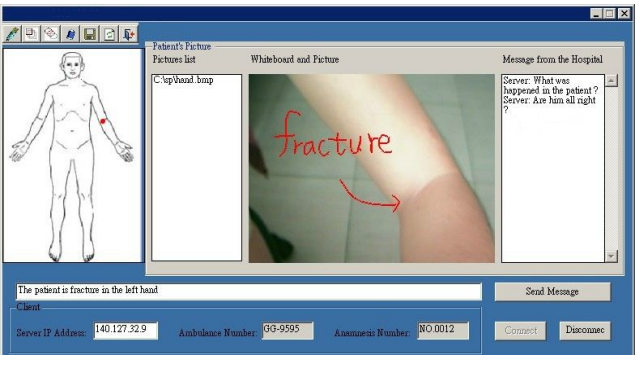


Fig. 4. The picture/whiteboard functions

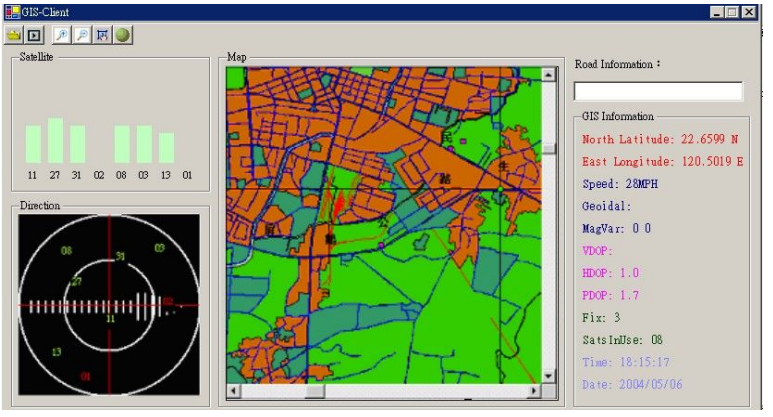


Fig. 5. The GIS on the ambulance

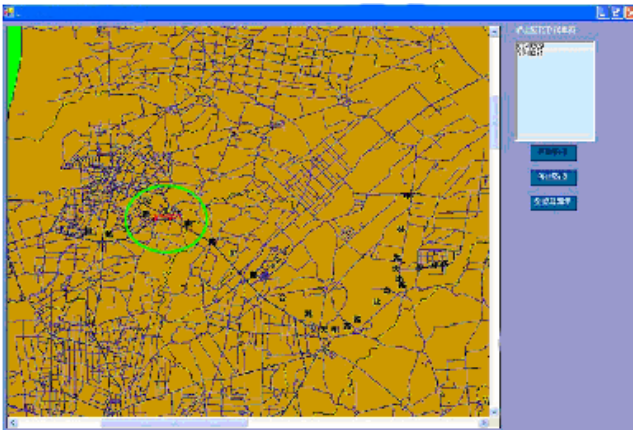


Fig. 6. The ambulance's locations at the Emergency Control Center

signal strength of satellites and the longitude and latitude of current location. When an accident occurs, the staff of the ambulance dispatch center can determine the appropriate ambulance to go to the accident using the functions depicted in Fig. 6. From the functions, the people can understand where the ambulances are and whether they have emergency tasks.

4 Conclusions

The rapidly increasing application of handheld devices and the deployment of mobile communication environment are accelerating the development of pervasive telemedicine services. The proposed CANE system tries to compensate the limitations of conventional emergency medical services, by which the EMT personnel merely

inform doctors about patient's symptoms by telephone. CANE improves the efficiency of the emergency remedy using advanced pervasive multimedia techniques. Physicians can make accurate diagnoses and recommend appropriate treatment suggestion to the EMT staff based on the context-aware information, including the real-time multimedia data about the patient's symptoms. The context-aware information helps physicians to diagnose illnesses and gives the hospital staff a head start in assessing the patient and preparing suitable treatment. Real-time pervasive access to the patient is one of the improvement factors in extending lives and lengthening the rescue time. To achieve practical usage in the near future, the researchers are cooperating with a local teaching hospital to refine and verify the effectiveness and value of the CANE system.

Many counties make much of emergency and use a diverse array of information technologies to assist emergency person to complete emergency tasks. However, many emergency staffs think that this approach increases their loads by adding extra equipment in emergency. Moreover, emergency care combining the state-of-art Information technology is required immediately before telemedicine becomes legal in Taiwan.

Acknowledgement

The authors would like to thank the National Science Council of the Republic of China, Taiwan for financially supporting this research under Contract No. NSC 94-2218-E-020 -001.

References

1. Arshad, U., Mascolo, C., and Mellor, M.: Exploiting mobile computing in health-care. Proceeding of the Third International Workshop on Smart Appliance, Vol. 1. (2003) 151-155
2. Miguel, M., Marcela, R., Jesus, F., Ana, I.M., and Victor, M.G.: Context-aware mobile communication in hospitals, Vol. 36. No. 9. IEEE Computer (2003) 38-46
3. Pattichis, S., Kyriacou, E., Voskarides, S., Pattichis, M.S., Lstepanian, R., and Schizas, C.N.: Wireless telemedicine systems: an overview, Vol. 44. No. 2. IEEE Antennas and Propagation Magazine (2002) 143-153
4. Satyanarayanan, M.: Pervasive computing: vision and challenges, Vol. 8. No.4. IEEE Personal Communications (2001) 10-17
5. Baker, B.: East meets west to cure healthcare ills using information technology. Proceedings of The Digital Hospital Medical Information Systems. (2004) 117-128
6. Choudhri, Kagal, L., Joshi, A., Finin, T., and Yesha, Y.: PatientService: electronic patient record redaction and delivery in pervasive environments. Proceedings of 5th International Workshop on Enterprise Networking and Computing in Healthcare Industry. (2003) 41-47
7. Znati, T.: On the challenges and opportunities of pervasive and ubiquitous computing in health care. Third IEEE International Conference on Pervasive Computing and Communications (PerCom 2005). (2005) 396-396
8. Basharat, Catbas, N., and Mubarak, S.: A framework for intelligent sensor network with video camera for structural health monitoring of bridges. Third IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom 2005). (2005) 385-389

9. Perry, M., Dowdall, A., Lines, L., and Hone, K.: Multimodal and ubiquitous computing systems: supporting independent-living older users, Vol. 8. No. 3. IEEE Transactions on Information Technology in Biomedicine (2004) 258-270
10. Abowd, G.D., Mynatt, E.G., and Rodden, T.: The human experience, Vol. 1. No. 1. IEEE Pervasive Computing (2002) 48-57
11. Anand R., Roy H.C., and Anupama M.: ConChat: A context-aware chat program, Vol. 1. No. 3. IEEE Pervasive Computing (2002) 51-57
12. Paolo, Antonio, C., and Cersare, S.: The ubiquitous provisioning of internet services to portable devices, Vol. 1. No. 3. IEEE Pervasive Computing (2002) 81-87
13. Miyamoto, M., Mako, M., Kimura, M., and Kanno, T.: Great eqrthquakes and medical information systems, Vol. 6. Journal of American Medical Informatics Association (JSMIA) (1999) 252-258
14. Delp, S.L., Loan, P., Basdogan, C., Rosen, J.M.: Surgical simulation: An emerging technology for training in emergency medicine, Vol. 6. No. 2. Journal of Virtual Environments in Medicine (1997) 11-29
15. Moore S.K.: Extending healthcare's reach, Vol. 39. No. 1. IEEE Spectrum (2002) 66–71
16. Razzak, J.A. and Kellermann, A.L.: Emergency medical care in developing countries: is it worthwhile, Vol. 80. No. 11. Bulletin of the World Health Organization (2002) 900-905
17. Derekenaris, G., Garofalakis, J., Makris, C., Prentzas, J., Sioutas, S., and Tsakalidis, A.: An information system for the effective management of ambulances. Proceedings of 13th IEEE Symposium on Computer-Based Medical Systems (CBMS'00) (2000) 269-274
18. Helal, S., Winkler, B., Lee, C., Kaddoura, Y., Ran, L., Giraldo, C., Kuchibhotla, S., and Mann, W.: Enabling location-aware pervasive computing applications for the elderly. Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom 2003) (2003) 531-536
19. Schepps, J. and Rosen, A.: Microwave industry outlook – wireless communications in healthcare, Vol. 50. No. 3. IEEE Transactions on Microwave Theory and Techniques (2002) 1044-1045
20. Lakshmi Narasimhan, V., Irfan, M., and Yefremov, M.: MedNet: a pervasive patient information network with decision support. Proceedings of 6th International Workshop on Enterprise Networking and Computing in Healthcare Industry. (2004) 96-101
21. Varshney, U.: Pervasive healthcare, Vol. 36. No. 12. IEEE Computer (2003) 138-140
22. Mynatt, D., Melenhorst, A.S., Fisk, A.D., and Rogers, W.A.: Aware technologies for aging in place: understanding user needs and attitudes, Vol. 3. No. 2. IEEE Pervasive Computing (2004) 36-41
23. Jakob: Hospitals of the future: ubiquitous computing support for medical work in hospitals. Proceeding of the 2nd International Workshop on Ubiquitous Computing for Pervasive Healthcare Applications (UbiHealth 2003) (2003) 171-175.

Design and Implementation of Interactive Contents Authoring Tool for MPEG-4

Hsu-Yang Kung, Che-I Wu, and Jiun-Ju Wei

Department of Management Information Systems,
National Pingtung University of Science and Technology, Pingtung, Taiwan
{kung, joey, m9156021}@mail.npust.edu.tw

Abstract. This paper proposed and implemented a QoS-Aware MPEG-4 authoring tool with provision of content interaction control, which is named the Q4AT system. Q4AT provides friendly user interface in a WYSWYG manner for users to create their favorite MPEG-4 scenes. The characteristics of Q4AT are as follows. (i) The description of a scene is based on the standard BIFS language. (ii) The creation of the object description is based on the syntactic description language (SDL). (iii) All descriptions are VRML-based. A VRML-based system is compatible to different browsers and operation systems. Therefore, the author could simplify the steps of creating an MPEG-4 scene. (iv) Q4AT supports the functions of the quality of service by the proposed object priority differentiation control. Using Q4AT, authors are transparent to the MPEG-4 processing layers and are capable of concentrating on the editing function usage of the Q4AT. Q4AT makes it possible to easily create a complicated MPEG-4 scene and reduce the difficulty of MPEG-4 media creation and presentation.

1 Introduction

MPEG-4 is a compression standard developed by ISO/IEC MPEG (Moving Picture Experts Group) and is totally different from MPEG-1/2. MPEG-4 standard improves the efficiency of media compression and tries to provide the applications integrating multimedia, communications, and computers [1]. The fundamental infrastructure of MPEG-4 is object-oriented and media operations (presentations) described in an MPEG-4 scene [2], [3]. An MPEG-4 scene combines the object descriptions of natural and synthetic media objects, e.g., the text, image, 3D, audio, and video objects, which can be manipulated by interaction operations [4], [5], [6]. To achieve a user-oriented MPEG-4 scene presentation, a user friendly MPEG-4 authoring tool is urgently required [7], [8]. Using a well-design authoring tool, a film maker could dynamically change the structure of an MPEG-4 scene to achieve the desired presentation contents [9], [10].

In the paper, we proposed and implemented a QoS-Aware MPEG-4 authoring tool with provision of content interaction control (Q4AT). Q4AT is developed using VC++, OpenGL, and VRML languages. *Q4AT* uses the standard BIFS language and the syntactic description language (SDL) to create the descriptions of a scene and the

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 785 – 794, 2005.
© IFIP International Federation for Information Processing 2005

media objects, respectively. All descriptions are VRML-based to be compatible to different browsers and operation systems. To support multimedia communications, *Q4AT* provides the QoS specifications via the proposed object priority differentiation control. *Q4AT* provides friendly user interface in a WYSWYG manner for users to create their favorite MPEG-4 scenes.

The rest of this paper is organized as follows. Section 2 describes the system architecture of Q4AT. Section 3 illustrates the implementation results and usage of Q4AT. Section 4 concludes this paper.

2 The System Architecture of Q4AT

Figure 1 depicts the system architecture of *Q4AT*. The processes of creating a standard MPEG-4 file (.mp4) are composed of authoring actions, e.g., the Edit actions, the Preview actions, and the Save actions. For example, an author news (opens) an mp4 file to edit, previews the authoring results using the VRML-enabled window, and saves the final results in the mp4 format [11]. The main components of the *Q4AT* system are (i) the Editing Board component and (ii) the Q4AT Function component, which are described as follows.

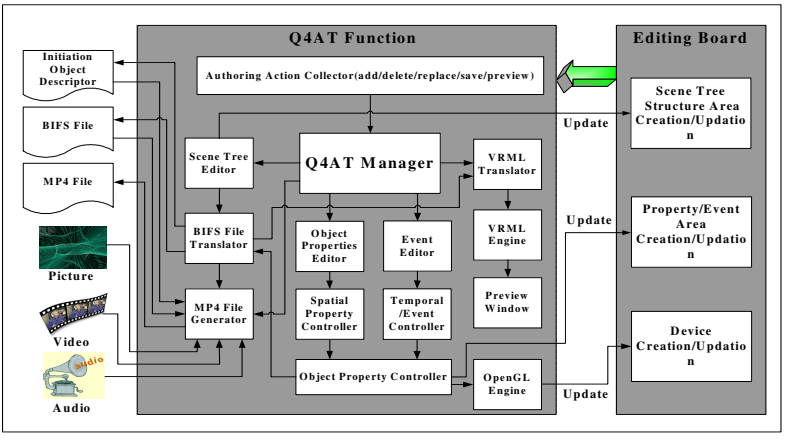


Fig. 1. The system architecture of the Q4AT system

2.1 Editing Board Component

Figure 2 illustrates the user interface, which is Editing Board component of *Q4AT*. There are five control modules composed of the editing component, which are (i) Pre-define Button, (ii) OpenGL Device, (iii) Property Area, (iv) Scene Tree Structure Area, and (v) Preview Window. We describe the functionality of control modules as follows.

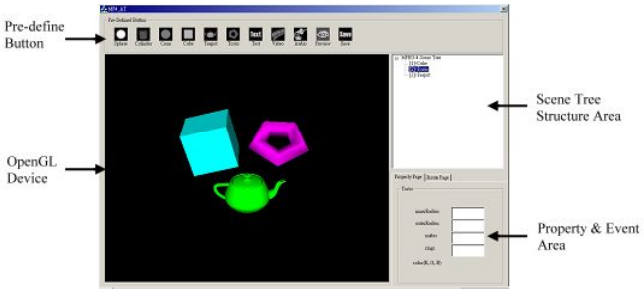


Fig. 2. The user interface of Q4AT

2.1.1 Pre-define Button

OpenGL Device is responsible for rendering an MPEG-4 scene in the WYSWYG way [12]. OpenGL Device translates and displays all editing actions of a scene author. Figure 3 illustrates the display result after editing media object using OpenGL Device. An author can change the coordination and view points of objects. When the properties of objects are changed, OpenGL Device re-displays the modified object at the same time.

(i). Object Button. Object Button provides the functionality for an MPEG-4 author to create various media objects via the defined icons. The icons of Object Button include Sphere, Cylinder, Cone, Cube, Teapot, Torus, Text, Video, and Audio. An author uses the icons of Object Button creates (editing) an complicated MPEG-4 scene, which is composed of the 2D 3D picture text audio, and video objects. Noted that there are corresponding video and audio files, which is specifies by the author, for the creation of video and audio objects. The scene author adds these objects to OpenGL Device and specifies the properties of media objects.

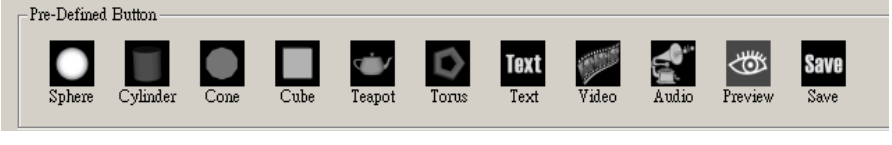


Fig. 3. Pre-define buttons of Q4AT

(ii). Preview Button. An author uses Preview Button to view the designed scene in advance of the next editing operation. The control policy of Pre-view Button is to trigger the VRML preview window and pop-up from Q4AT.

(iii). Save Button. An author uses Save Button to save the editing results of the designed MPEG-4 scene. When Save Button is clicked, MPEG-4 File Generator activates to collect attribute information of objects and MPEG-4 File Generator packs the related object files into a standard .mp4 file.

2.1.2 OpenGL Device

OpenGL Device translates and displays all editing actions of a scene author. Figure 4 illustrates the display result after editing media object using OpenGL Device. An author can change the coordination and view points of objects. When the properties of objects are changed, OpenGL Device re-displays the modified object at the same time.

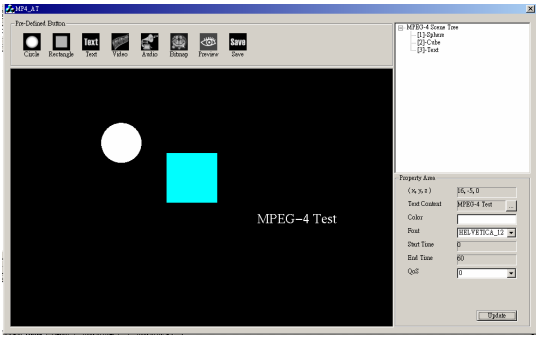


Fig. 4. A display example using OpenGL Device

2.1.3 Property Area

Property Area provides different objects of an MPEG-4 scene with the assignment of object attributes. Different objects may have same properties, e.g., color assignment, spatial/temporal values, and QoS priority, and also have heterogeneous properties, e.g., text font, encoding types of audio/video files, and sphere radius. The Property Area displays the designed object properties and gives the author the limitation of authority for editing different kind of media objects. OpenGL Device directly renders the modification of objects. Figure 5-(a) and -(b) illustrates a dialog example of using Property Area for specifying the attributes of a sphere object and a text object, respectively. In Property Area, some fields for specifying an object are changeable and some files are not. An author can modify the attribute files of a sphere object, including innerRadius, outerRadius, insides, rings, color and QoS.

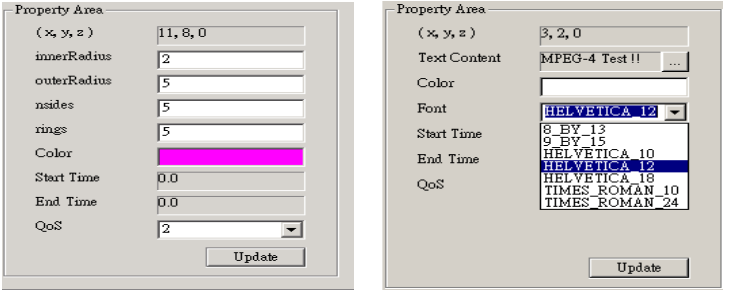


Fig. 5. The dialog windows of Property Area for (a) sphere object and (b) text object

2.1.4 Preview Window

Scene Tree Structure Area displays the tree structure of an MPEG-4 scene, which illustrates the detailed information about media objects, including the object id, the object name, the object type, and the temporal/spatial relationship between other media objects. Figure 6 depicts the functions of Scene Tree Structure Area and is described as follow.

(i). As an author uses Pre-define Button to add a medium object on OpenGL Device, Scene Tree Structure Area creates the corresponding tree node. A node of the scene tree contains the object information, e.g., object ID and object type.

(ii). When an author clicks a node of Scene Tree Structure Are, Property Area displays its properties to set attribute values of the node, i.e., the medium object.

(iii). An author can drag a node of scene tree to change the relationship between other nodes. To ensure the attribute consistence between tree nodes, Q4AT automatically monitors the legality of scene objects to prevent some violation situations.

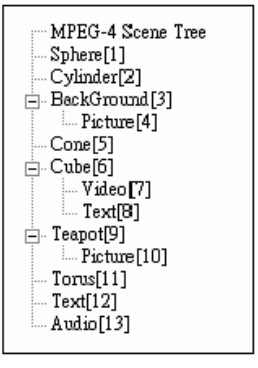


Fig. 6. An illustration example of Scene Tree Structure Area

2.1.5 Preview Window

Preview Window is a pop-up window for scene authors to preview the results of scene realization. Preview Window was manipulated via VRML. Using VRML, the problem that OpenGL Device can't integrate with the video objects is solved. Furthermore, VRML supports the capability of content interaction. A scene author is capable of dynamically and conveniently adjusting the status of a scene in a WYSWYG manner.

2.2 Q4AT Function Component

The Q4AT Function component is responsible for executing and realizing the authoring actions. All authoring actions can be considered as the transmission of action messages between Q4AT Function component and the author i.e., Editing Board component. The Q4AT Function is mainly composed of the Authoring Action Collector, Object Property Controller, BIFS File Translator, VRML Translator, and MP4 File Generator, which provide an author with a transparent environment to easily create a desired MPEG-4 presentation. We describe the main control modules as follows.

(i). Authoring Action Collector gathers the editing actions of a scene author and sends editing actions to Q4AT Manager. Q4AT Manager distributes actions messages to the related Editors to manipulate media object.
(ii). Object Property Controller gets action messages, including the adding, deleting, properties modification, refreshing actions, from the Q4AT Manager. Then, Object Property Controller transmits the gathering messages to OpenGL Device and BIFS File Translator for the displaying and translation of a scene.
(iii). BIFS File Translator detects changes of object properties and keeps up the latest properties prepared for VRML Translator and MP4 File Generator. The main purpose of BIFS File Translator is to create the scene description and object description of MPEG-4 standard, and then generates the BIFS file.
(iv). VRML Translator gets the BIFS file from BIFS File Translator and translates the BIFS file into the formats of VRML syntax used for Preview Window.
(v). The main function of MP4 File Generator is to build the standard MPEG-4 file (.mp4). MP4 File Generator gathers all media files of the corresponding media objects and packs the media files with scene/object description into a standard MPEG-4 file.

3 Usage of Q4AT

The steps of realizing an MPEG-4 scene are to add media objects into a scene, to modify the properties of objects, to preview the authoring results, and to finally save the editing results. Figure 7 illustrates an example of editing an MPEG-4 scene using Q4AT. Figure 7-(a) shows the authoring results of adding two objects into OpenGL Device. A scene author uses mouse to easily drag media objects in a scene and to change their spatial attributes or viewpoint. An author uses Scene Tree Structure Area to add a corresponding node at the same time. There are two ways to modify the properties of a medium object. (i) An author can use the mouse to click the object in OpenGL Device. Or, (ii) an author clicks the node of Scene Tree Structure Area to modify the properties of the node. These two actions will trigger the Object Property Area to display corresponding properties of nodes, and then scene author modifies the changeable attributes in Object Property Area and change the attribute values directly. Note that not all properties in Object Property Area are modifiable, for example, the spatial value of a medium object is unchangeable. Figure 7-(b) depicts the color dialog for modifying the colors of an object. Figure 7-(c) shows the timeline dialog for editing the start/end time of a medium object. OpenGL Device displays all modifications in real-time for the scene author.

Due to the critical limitation of integrating video objects by using OpenGL, Q4AT uses a 2-D rectangle to express a video object and provides VRML Preview Window to solve the problem of OpenGL. Furthermore, the designed interaction functions are useful and convenient for scene authors to adjust the deployment of scene objects by using VRML. Figure 79-(d) shows the display example of a video object in OpenGL Device. Figure 7-(e) illustrates the preview result of using VRML Preview Window to preview the deployment of scene objects.

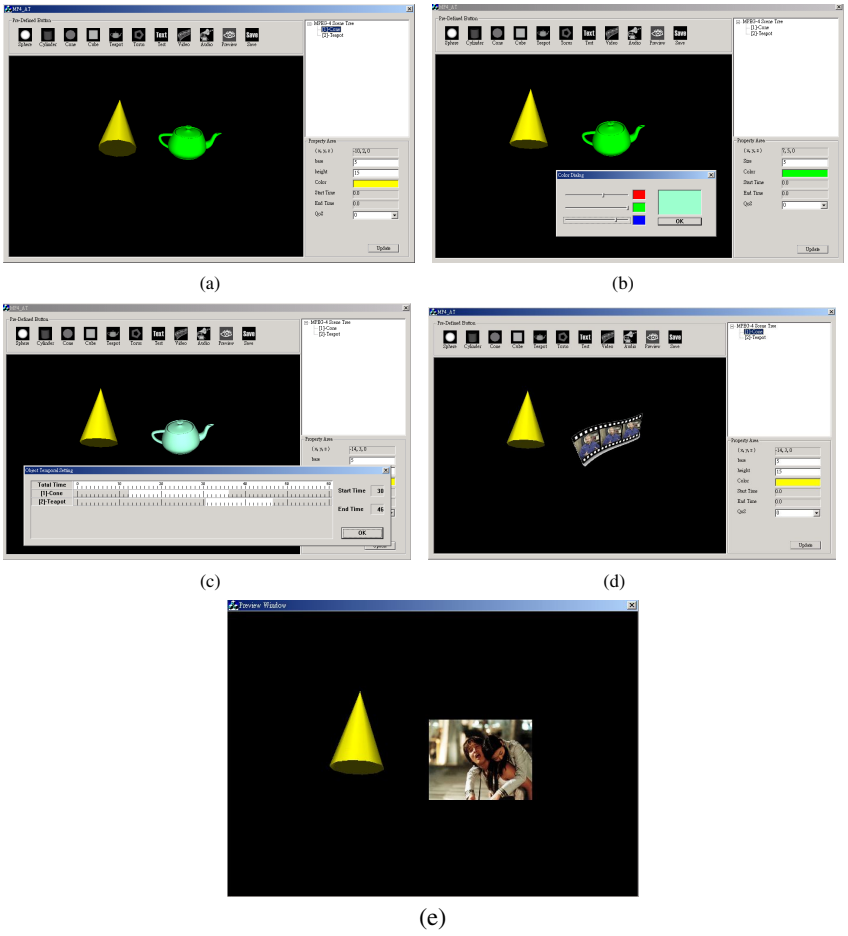


Fig. 7. (a) Adding two media objects. (b) A Color dialog. (c) A TimeLine dialog. (d) Adding video object into a scene. (e) Using VRML Window to preview.

After authoring the MPEG-4 scene, the BIFS File Translator of Q4AT translates the designed scene tree into the scene.bt file, which contains the standard BIFS commands with text format. Q4AT adopts the MP4Box encoder [13] to interpret the BIFS commands in the scene.bt file and then encode the BIFS commands into the scene.mp4 file. The scene.mp4 file is just the scene and object description file with binary format. The related media content files are pointed in the scene.mp4. Figure 8 depicts the processing procedure of generating a recognized .mp4 file and the processing steps are as follows.

(i). The acceptable medium format for the video objects is the .media format. Firstly, Q4AT adopts the DivxEnc converter [14] to translate the original medium file with .avi format into the intermediate file with .m4v format.

Secondly, Q4AT adopts the MP4Creater60 converter [15] to translate the intermediate medium file from the .m4v format into the .mp4 format. However, the .mp4 format file created by MP4Creator60 is only the MPEG-4 simple profile format, which can't be used for the MPED-4 core profile environment, i.e., the multiple object authoring and presentation environment. Therefore, Q4AT adopts the MP4Box encoder to translate the .mp4 file into there files with the .nhnt, .info, and .media formats. The generated medium file with .media format can be realized by MP4Box for video and audio objects.

(ii). The acceptable medium format for the audio objects is the .media or .mp3 format.

(iii). The acceptable medium format for the image objects is the .jpg or .bmp format.

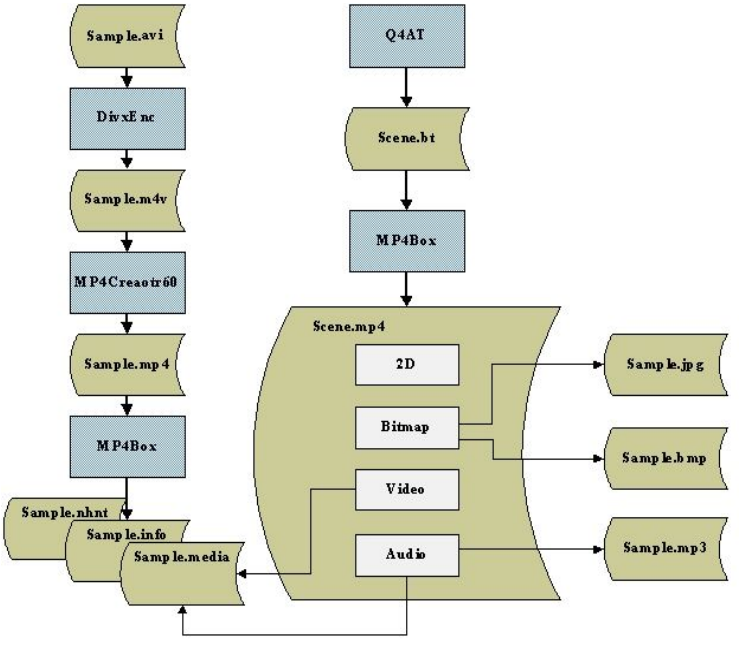
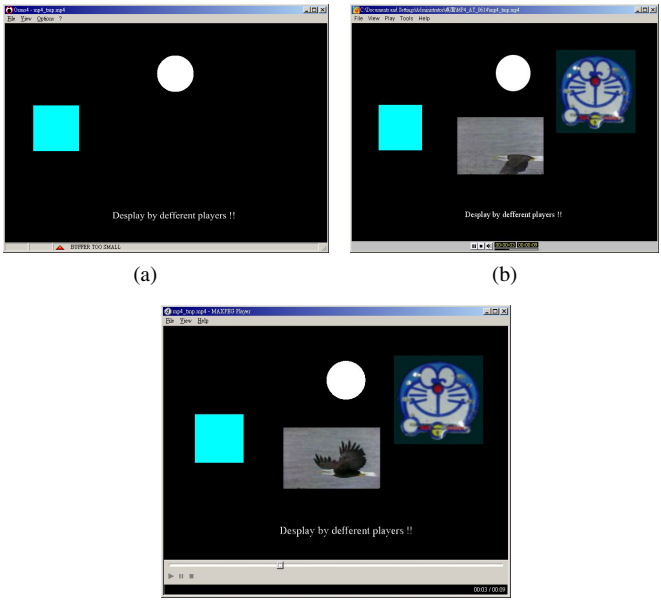


Fig. 8. The processing procedure of generating a .mp4 file

To prove the feasibility of Q4AT, this paper uses the well-known MPEG-4 players, including the Osmo4 of ENST [16], the M4Play of IBM [17], and the MAXPEG of DigiMax [18], to present the generated .mp4 file. Figure 9-(a), -(b), and -(c) depicts an MPEG-4 file presentation results, which is authored by Q4AT, by using the Osmo4 player, the M4Play player, and the MAXPEG player, respectively. Figure 10 depicts the designed MPEG-4 scene presented on a PDA. The MPEG-4 player is the Osmo4, which is WinCE-based and is the only available MPEG-4 player for handheld devices.



(a) (b)

(c)

Fig. 9. The MPEG-4 presentation illustrations using (a) Osmo4 player, (b) M4Play player, and (c) the MAXPEG player

Fig. 10. The MPEG-4 presentation illustration on a PDA

4 Conclusion

In the paper, we proposed and implemented a QoS-Aware MPEG-4 authoring tool with provision of content interaction control (*Q4AT*). The *Q4AT* system provides the friendly user interface and helps the author easily create an MPEG-4 scene composed of heterogeneous media objects. The capabilities of the QoS-aware with the object priority control and the content interaction with the interaction event control are

provided in the *Q4AT* system. To achieve a smooth MPEG-4 presentation, the *Q4AT* system designs Object Property Area to specify the temporal/spatial relations of the media objects. The main goal of *Q4AT* is to provide powerful functionality for authors, who don't need have the complicated MPEG-4 knowledge, are capable of creating a standard MPEG-4 file.

Acknowledgement

The authors would like to thank the National Science Council of the Republic of China, Taiwan for financially supporting this research under Contract No. NSC-93-2219-E-020-002.

Reference

1. ISO/IEC 14496-1: 1999 Information technology -- Coding of audio-visual objects -- Part 1: Systems (1999)
2. Eleftheriadis, A.: MPEG-4 systems: architecting object-based audio-visual content. IEEE Second Workshop on Multimedia Signal Processing (1998) 535-540
3. Herpel, C.: Elementary stream management in MPEG-4, Vol. 9, Issue 2. IEEE Transactions on Circuits and Systems for Video Technology (1999) 315-324
4. Viljoen, D. W., Calit,z A. P., Cowley, N. L. O.: A 2-D MPEG-4 multimedia authoring tool. Proceedings of the 2nd international conference on Computer graphics, virtual Reality, visualization and interaction (2003) 151-160
5. Kim, K. Y., Hong, J. W., Park, C. M., and Jung, H. K.: The architecture of MPEG-4 based IPMP authoring system. Proceedings of IEEE International Conference on Multimedia and Expo (ICME '04) Vol. 1. (2004) 447 - 450
6. Liu, D. R., Shieh, M. J., Lee, Y. C., and Chen, W. C.: On the Design and Implementation of an MPEG-4 Scene Editor. Digest of Technical Papers of International Conference on Consumer Electronics (2000) 120-121
7. Bulterman, D. C. A. and Hardman, L.: Structured multimedia authoring, Vol. 1. Issue 1. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP) (2005) 89-109
8. Pourmohammadi-Fallah, Y., Asrar-Haghighi, K., and Alnuweiri, H. M.: Streaming multimedia over the Internet, Vol. 23. Issue 1. IEEE Potentials (2004) 34-37
9. Daras, P., Kompatsiaris, I., Raptis, T., and Strintzis, M. G.: An MPEG-4 tool for composing 3D scenes, Vol. 11. Issue 2. IEEE Multimedia (2004) 58-97
10. Drumm, H. and Kuhhirt, U.: The shadow tool - new visual cues for MPEG-4 authors. Proceeding of 2004 IEEE International Symposium on Consumer Electronics (2004) 77 - 82
11. Kim, K., Lee, I., and Ki, M.: Interactive contents authoring system based on XMT and BIFS. Proceedings of the tenth ACM international conference on Multimedia (2002) 275-278
12. Zhigen, X., Yan, Y., and Chen, J. X.: OpenGL programming in Java, Vol. 7. Issue 1. Computing in Science and Engineering (2005) 51 - 55
13. MP4box, http://www.aziendeassociate.it/cd/gpac/dev/MP4Box.7z
14. DivxEnc, http://www.greatis.com/security/download.htm
15. MP4Creator60, http://www.esato.com/archive/t.php/t-63404
16. Osmo4, http://www.comelec.enst.fr/osmo4/
17. M4Play, http://www.alphaworks.ibm.com/tech/tk4mpeg4
18. MAXPEG, http://www.digimax.com.tw/MPEG4/

A Programmable Context Interface to Build a Context Infrastructure for Worldwide Smart Applications

Kyung-Lang Park, Chang-Soon Kim, Chang-Duk Kang, and Shin-Dug Kim

Super Computing Lab, Department of Computer Science,
Yonsei University, 120-749, 134 Shinchon-Dong,
Seodaemoon-Gu, Seoul, Korea
{lanx, flsoon, niceguy, sdkim}@yonsei.ac.kr
http://supercom.yonsei.ac.kr

Abstract. Context-awareness is one of the most important technologies for the ubiquitous computing. To embed such a technology into applications, first of all, a well-defined architectural framework is required to support applications to obtain necessary contexts. Several studies have been proposed, but most of them overlooked one fundamental feature that contexts are subjective things and context-awareness is subjective behavior. Thus, providing pre-defined, pre-programmed operators which generate contexts by combining a couple of sensed data is very impractical. Instead, in this paper, we provide a well-defined programmable interface to applications, so that they can obtain and use contexts according to their own ideas. To evaluate our architecture and context interface, we implement all components comprising of the context infrastructure and perform several experiments. The results show the proposed method provides a flexible and expressive interface, but does not countervail the performance.

1 Introduction

Ubiquitous computing [11] is being considered as a next generation computing paradigm these days. Computer-based technologies will be embedded into our casual life ubiquitously, so that we can use a variety of smart services anytime and anywhere. One of the most important technologies of the ubiquitous computing is context-awareness. If a service perceives our situational contexts, it will operate much smarter without any artificial intelligence stuff.

The most fundamental idea is to insert various sensors into an application. Then, the application can be aware of our situation by analyzing sensed data. However, such a sensor-driven approach gives developers pain because the developers have to know about all the sensors in detail. They have to spend more time on sensors than the core logics of applications. Several approaches have been proposed to overcome the problem. They attempted to separate context handling modules from applications and provided convenient toolkits and frameworks to develop sensor-driven smart applications much easier [1, 2, 7, 15, 16]. However, they are not sufficient enough to support the development of enormous context-aware applications. Nowadays, thus, an approach to establish a context infrastructure for innumerable context-aware applica-

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 795 – 804, 2005.
© IFIP International Federation for Information Processing 2005

tions is on the rise. When a context infrastructure is constructed, we can share a lot of sensors, information sources, and processing components. Consequently, we will design smart applications without any difficulty [8].

Some studies are based on the infrastructure approach [4, 12, 13]. They assume a worldwide model and consider various and enormous context-aware applications which use shared sensors in the world. Although they addressed some infrastructure issues such as scaling up, information dissemination, and unified sensor abstraction, they overlooked one important feature that contexts are subjective things rather than objective. To generate a context, several sensed data have to be aggregated. However, the problem is that the developers of the context infrastructure and developers of applications are different. Even though a well-defined component in the context infrastructure aggregates sensed data and generates contexts with a novel algorithm, they cannot be accepted to some applications. Thus, it is required to support applications by specifying their ideas to generate and use contexts.

In this paper, we propose an architecture for the context infrastructure and describe how applications obtain and use contexts subjectively. Our context interface provides a context description language (CDL) for applications to specify their requests. Thus, based on the CDL given by the application, the infrastructure discovers proper sensors, aggregates sensed data, generates contexts, and sends them to the applications. To evaluate our architecture and context interface, we implemented all components comprising of the context infrastructure and did several experiments. Experimental results show that the proposed method provides a flexible and expressive interface, but does not countervail the performance. Rest of the paper is organized as follows: In Section 2, we introduce several related work. Section 3 describes our context infrastructure and Section 4 explains the programmable context interface which is the core of our infrastructure. Section 5 provides several experimental results to evaluate our research. Finally, we conclude in Section 6.

2 Related Work

There are several studies to support development of context-aware applications. The Context Toolkit [1] is a pioneering work of this area. It suggested a well-defined conceptual framework that supports context-aware applications and provided several software components which aggregate and manage sensed data. Thus, developers can make context-aware applications rapidly by using the Context Toolkit. Since the advent of the Context Toolkit, a number of researches have been appearing. [2, 5, 7, 9, 15, 16].

Those are very convenient to develop a few number of applications in a domain, but not sufficient to support a large number of applications in multiple domains. Hong's article pointed out such a problem and introduced an infrastructure approach to develop a large number of context-aware applications [8]. There are several projects based on the infrastructure approach. Chen et al. proposed Solar, an infrastructure for context-aware applications [4]. Solar generates contexts by using an operator graph, where an operator is a self-contained data-processing component. A higher-

level operator can be made by composing existing operators. Because Solar considered a large-scale context fusion network, it addressed several infrastructure issues, scalability and addressing. Contextor Infrastructure [6] took up the similar fashion. It suggested contextors as a building block of the infrastructure, where a contextor is a component which can provide contextual information. Either a sensor or a processing unit like an operator of Solar can be a contextor. They suggested applications retrieved contexts by composing several contextors. SCINet (Strathclyde Context Infrastructure) [13] and NEXUS [12] designed local context servers and extended it to be used for global areas by composing network of servers. SCINet comprises of CSs (Context Server) which maintains a central store of contexts and provides the access point for the applications. When a CS receives a query from an application, a query resolver in the CS generates a path to make the required context. In NEXUS, they defined an AHSS (Aware Home Spatial Service), a kind of context server, which is suitable for mid-size home environments and bridged AHSSs by using a query interface named AWQL. Semantic Space [17] and Gaia's context infrastructure [3] are also aimed to construct an infrastructure for context-aware applications. Semantic Space used context wrappers as a building block of the infrastructure. It also leveraged technologies from the Semantic Web. It provides RDF-based query language to combine sensors and existing semantic web information. Gaia project also assumed that there would be a lot of context providers and defined how context providers generate contexts.

Our work has the same perspective with above researches which are based on the infrastructure approach, but we do not agree with them in terms of context fusion and context inference in the infrastructure. We do not allow any kind of data composition predefined in the infrastructure. Instead, we provide more flexible and programmable interface to applications, so that they can obtain and use subjective contexts.

3 Architecture for a Context Infrastructure

Most of all, a well-defined architecture is required to build a context infrastructure for worldwide smart applications. In infrastructure approach, sensors are not for a specific application. Thus, we should define a bottleneck protocol to glue sensors onto an application. Figure 1 shows our architecture for the context infrastructure. We divide the bottleneck protocol into two layers. One is *abstraction layer* and the other is *collective layer*. The abstraction layer hides complexity of the actual sensors by wrapping them with a single software interface and publishes sensors to the outside. Typically, sensors use different I/O interfaces and express sensed data by using different data models and formats. Thus, it is quite difficult for an application to understand several kinds of I/O interfaces, data models, and formats. When sensors are abstracted, upper-layer components can access to the sensors through a single interface without deep knowledge of each sensor.

Our virtual sensor is based on Web services. It is based on fully open and standard protocols and does not depend on the physical communication method. A virtual

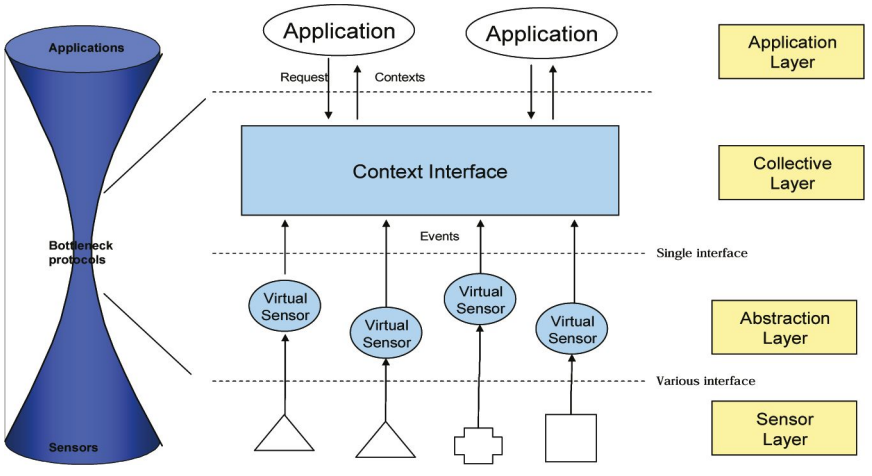
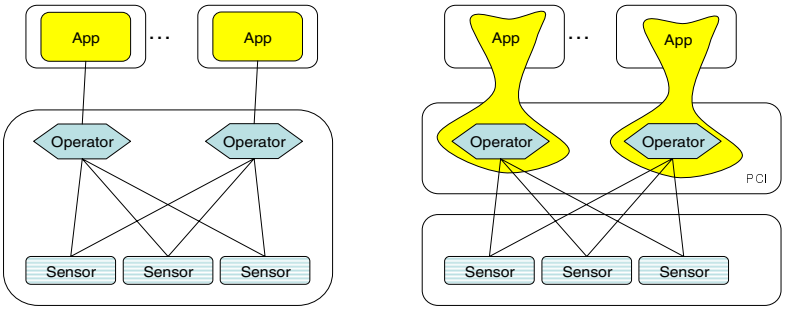


Fig. 1. The proposed architecture for the context infrastructure

sensor expresses sensed data as a tuple which comprises of *entity*, *location*, *time*, *attribute*, and *value*. The value is actual sensed data and others are kind of meta-information to help applications to understand the value. Upper-leyer components have only to know the Web service interface and the tuple model of the virtual sensor to access various sensors.

However, abstraction layer is not sufficient to support context-aware applications. First of all, applications do not know where proper sensors are. Thus, discovery service has to be provided to applications. Also, most of sensors are immature. They provide imperfect information and lack the confidence of information. Thus, they have to be combined to provide necessary information to applications. Both functionalities, i.e., sensor fusion and sensor discovery, are provided by the collective layer. It receives a request for contexts from an application, aggregates information from published virtual sensors, generates contexts by analyzing the information, and sends the contexts to the application. At this point, we should remind that contexts are subjective representation of information and context-awareness is a subjective behavior. If the collective layer generates contexts with its own algorithm, the contexts are not subjective products of the application but of the collective layer. Therefore, contexts can be used to only a few applications which agree with them. When we make a single application, it does not matter, but it cannot be used for the context infrastructure. Thus, it is more efficient to provide a method for applications to specify how to aggregate and generate necessary contexts. It is the programmable context interface (PCI) proposed in this paper. Figure 2 shows the difference between two. When the collective layer provides predefined operators, they are not a part of an application. Applications have to decide whether they use the operators or not. When using the PCI, operators can be a part of applications. Thus, applications can generate their own contexts subjectively.



(a) Predefined operators (b) the proposed PCI model

Fig. 2. Predefined operators vs. the PCI model

4 Programmable Context Interface

As pointed out in the previous sections, the core of the context infrastructure is the programmable context interface. By using it, applications can generate and utilize contexts subjectively. In this section, we describe the PCI and its components in detail.

It consists of three major components: application context managers (ACMs), the context broker (CB), and the channel manager (CM). The ACM is responsible for managing contexts for an application. At first, an application should generate a CDL to specify its request and submit to the context interface (1). The ACM factory receives the CDL and creates an ACM for the application (2). The ACM parses the CDL and constructs a tree structure based on the CDL (3). The CDL tree collects events from sensors, generates contexts, and sends them to the application. In the CDL, a sensor is specified by using either an end-point address of the sensor or a query to discover the sensor. If a query is used to describe a sensor, it will be sent to the CB after the ACM constructs the tree (4). Then, the CB discovers a proper sensor and sends the end-point address of the sensor to the ACM (5). When this stage is over, all sensor descriptions are filled with the specific addresses of sensors. It is called the grounded CDL. After that, the ACM sends the grounded CDL to the CM (6). The CM subscribes virtual sensors instead of the ACM (7). When an event from a virtual sensor arrives at the CM (8), it sends the event to the ACM (9). The CDL tree in the ACM receives the event and operates it according to the operation defined in the CDL (10). This operation flow is shown in Figure 3.

The specification of the CDL is the most important part of the PCI. By using the CDL, application can specify a lot of things to generate contexts by using the CDL. They can specify which sensors they use, how aggregate sensed data, and how generate the context from the sensed data. Thus, it has to be simple, but expressive enough to specify any operation required for applications. Figure 4 shows an example of the CDL.

A CDL comprises of three parts, a list of sensors, a list of operators, and a list of requests. <cdl:sensorList> specifies sensors which are going to be used. Sensors

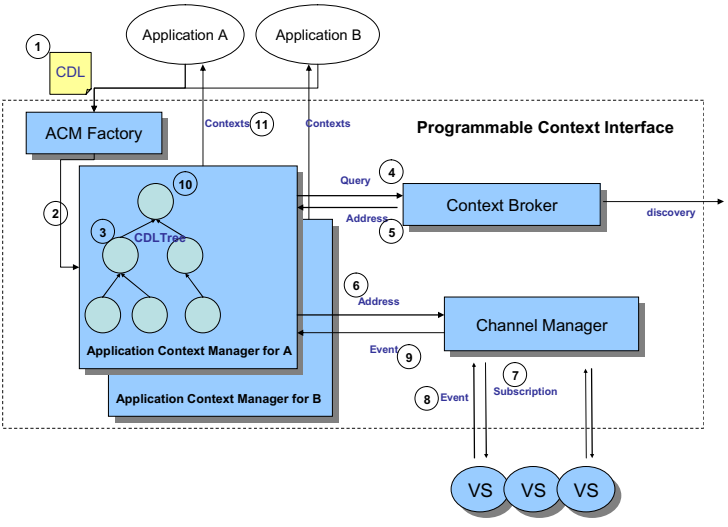


Fig. 3. Operation flow of the Programmable Context Interface

```
<?xml version="1.0" encoding="UTF-8"?>
<cdl:Definitions xmlnscdl="http://supercom.yonsei.ac.kr/2005/03/cdl090" name="Park Presence">
 <cdl:SensorList>
  <cdl:Sensor type="constant" name="S_ONE">
    <cdl:Query num="1">
                <cdl:Tuple>
                  <cdl:Location/>Z1</cdl:Location>
                  <cdl:attribute ontologyRef="http://supercom…/2005/03/ont/test/Pressure">Pressure</cdl:Attribute>
                </cdl:Tuple>
    </cdl:Query>
    <cdl:Parents num="1">
      <cdl:Parent name="OP_ONE"/>
    </cdl:Parents>
  </cdl:Sensor>
  <cdl:Sensor type="subscription" name="S_TWO">
    <cdl:Address>
      <cdl:URI>http://121.141.10.199/WS/Sensors/ID_Sensor.wsdl</URI>
      <cdl:Parameter/>
    </cdl:Address>
    <cdl:Parents num="1">
      <cdl:Parent name="OP_ONE"/>
    </cdl:Parents>
  </cdl:Sensor>
 </cdl:SensorList>
 <cdl:OperatorList>
   <cdl:Operator name="OP_ONE">
    <cdl:Inputs num="3">
                <cdl:Input name="S_ONE" driver="true"/>
                <cdl:Input name="S_TWO"/>
    </cdl:Inputs>
    <cdl:Operation type="mapping">
                <cdl:Mapping>
                        <cdl:Condition>
                        <cdl:Cmd>$S_ONE.value > 1 </cdl:Cmd>
                        <cdl:Cmd>$S_TWO.value == Park</cdl:Cmd>
                        <cdl:Tuple>
                                <cdl:Entity>Park</cdl:Entity>
                                <cdl:Location>Z1</cdl:Location>
                                <cdl:Time type="new"/>
                                <cdl:Attribute>Presence</cdl:Attribute>
                                <cdl:Value>true</cdl:Value>
                        </cdl:Tuple>
                        </cdl:Condition>
                        <cdl:Condition>
                        <cdl:Tuple>
                                <cdl:Entity>Park</cdl:Entity>
                                <cdl:Location>Z1</cdl:Location>
                                <cdl:Time type="new"/>
                                <cdl:Attribute>Presence</cdl:Attribute>
                                <cdl:Value>false</cdl:Value>
                        </cdl:Tuple>
                        </cdl:Condition>
                </cdl:Mapping>
    </cdl:Operation>
   </cdl:Operator>
 </cdl:OperatorList>
 <cdl:RequestList>
                <cdl:Request type="operator" name="OP_ONE"/>
 </cdl:RequestList>
</cdl:Definitions>/
```

Fig. 4. An example of the CDL. It represents a kind of person presence sensor.

can be categorized into two types, i.e., subscription and constant. The constant means that the sensor retrieves sensed data only once, but the subscription means that the sensor retrieves data continuously. A sensor can be specified by either <cdl:query> or <cdl:address>. When the application already knows the sensor, <cdl:address> is used. If the application does not know about the sensor, <cdl:query> has to be used to

discover a group of proper sensors. The query is based on the context tuple we defined. The query element is sent to the context broker (CB) and the CB transforms the query element into an address element. In this example, we specified two sensors. One is specified by a query element. The other is specified by an address element.

<cdl:operatorList> specifies operators. An operator is a kind of processing module. It receives data from sensors or other operators and processes the data with own operation. It can be defined by <cdl:inputs> and <cdl:operation>. The <cdl:inputs> specifies sensors and operators that have to send data to the operator. Only input elements marked as the driver can trigger the operation. The operation in an operator is specified by the <cdl:operation>. Current version of the CDL supports five types of operations. Those are mapping, selection, integration, calculation, and manual. Finally, <cdl:Request> is used to mark sensors or operations whose output would be sent to the application. When all nodes marked as request finish their operation, the ACM makes a result by combining events generated from requested nodes and sends it to the application.

5 Evaluation

In this section, we evaluate the proposed context infrastructure based on the programmable context interface. The main difference between previous approaches and our approach is that we do not use pre-defined operations. Instead, we provide an interface to applications to program operations that they need to obtain necessary contexts. Thus, we have to show that our operations can be performed in the same way as the pre-defined and pre-programmed operations.

In our approach, an ACM should construct a CDL tree on the fly. If there is heavy overhead to construct a tree, it cannot be used even though it provides a very flexible and convenient interface. This is because contexts are time-sensitive information. Following experimental results show the construction of a CDL tree does not take much time although there are a number of nodes specified in a CDL. They also show that overall response time of the PCI is as fast as other systems which use predefined, pre-programmed model.

5.1 Experimental Results

All components comprising of the context infrastructure are implemented by using Java SDK 1.5 and WSDP (Web Services Development Package) on Window XP machines. ACMs, the CB, the CM are located in the same machine, and each sensor and application are placed in different machines. A machine with the PCI components is Pentium IV 1.33GHz with 1GB main memory. All machines are connected by local area networks. We firstly make a CDL which consists of two sensors and one operator, and measure the time at several points to see the basic timing diagram. Figure 5 shows the result of the experiment. It takes about 400ms from an application submitted a CDL to the application received the result. Specifically, such a time delay is quite feasible because valid time period of a sensed data is generally a couple of seconds. The time from application launched to the time the ACM received the CDL stands for the communication time between two machines. It is not related to the PCI and depends on the network status. From the time to the CDL tree constructed stands

for the tree construction time. It is very important to evaluate the ACM. Thus, we will show other experimental results about the construction time. The next one is the time to discover sensors. It depends on the ability of the CB. The next is the time to subscribe to the sensors. It is very simple operation and depends on the network status rather than the ability of the CM. The time from all events arrived to the result stands for the processing time. It is also very important. The operator generated in the ACM interprets the operation specified in the CDL on the fly. Thus, it takes much more time compared to the pre-programmed operations. Finally, the time for the application to receive the result is also a kind of communication time between the PCI host and the application.

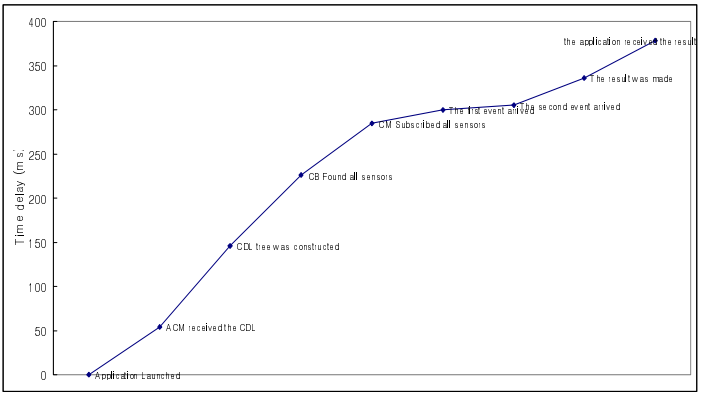


Fig. 5. The timing diagram when an application submits a CDL consisting of two sensors and one operation

The most important point in this time diagram is the construction time, because our approach generates operation trees in running time, which may not exist in other approaches. To measure the construction time, we perform additional experiments. We measure only construction time with changing the number of nodes. Figure 6 shows the result of these experiments. When the number of nodes is two, it takes only 97ms. It takes more time as the number of nodes increases. When the number of nodes becomes more than one hundred, the construction time will be less than one second. However, one CDL include about 10 to 20 sensors typically. It takes only 120ms, so that it does not cause any heavy overhead to the application.

Finally, four context-aware applications are designed by using several context acquisition models. The first model is pre-programmed and pre-defined model. In this model, operators are already developed and published, and applications already know about the operators. Probably, the programmer of operators and the programmer of applications should be same. The Context Toolkit uses this approach. The next one is pre-programmed and not-defined model. Several useful operators are provided by some participants, but the application should discover the operators. A lot of approaches based on the infrastructure such as Solar and GAIA use this model. Our PCI model is not-programmed and not-defined model. There is no pre-programmed

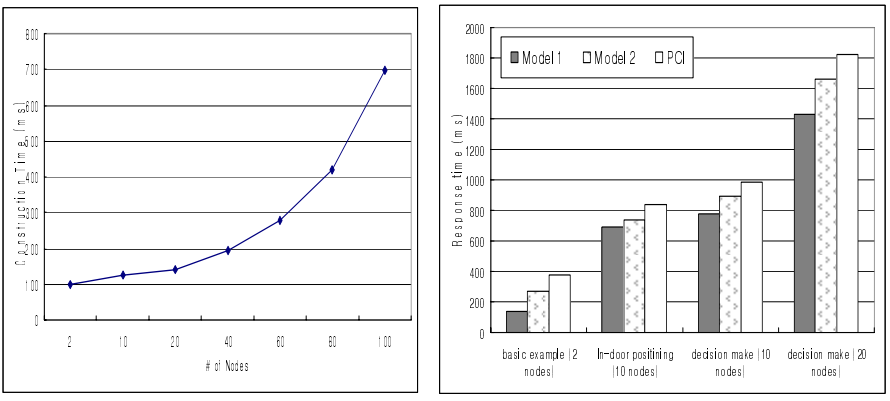


Fig. 6. CDL tree construction time as changing the number of nodes (left). Response time compared to other models (right).

operator. Instead, applications have to generate subjective operators by submitting CDLs. We compare the response time of three models. Definitely, the first model shows the best performance as shown in Figure 6. It is because there is no additional overhead in the model such as construction time and discovery time. The second model only has discovery time as overhead. However, there is no huge difference between three models in terms of the performance. Simply, it means that the construction time and discovery time occupy a small portion of the response time. But, the PCI model provides more convenient and flexible interfaces than other models. Thus, it is more feasible approach to support a number of context-aware applications.

6 Conclusion

In this paper, we have presented an architecture for supporting context-aware applications based on the infrastructure approach. The contribution of this paper is that we suggest a method that applications obtain and use contexts subjectively. When an infrastructure provides predefined and pre-programmed operators to extract contexts by using sensed data, it cannot be used for many applications. It is more efficient to provide a programmable interface to applications to specify their requests for contexts. Experimental results show that it provides feasible response time compared to other context acquisition models.

References

1. A. K. Dey, D. Salber, and G. D. Abowd,: A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications, Human Computing Interaction. Vol. 16. (2001) 97-166
2. A. Schemidt, U. Karlsruhe, K. V. Laerhoven,: How to build Smart Applications, IEEE Personal Communications. (2001)

3. A. Ranganathan and R. H. Campbell.: An infrastructure for Context-awareness based on first order logic, Personal and Ubiquitous Computing, Vol. 7. Springer-Verglag London (2003) 353-364
4. G. Chen, M. Li, and D. Kotz.: Design and Implementation of a Large-Scale Context Fusion Network. In Proc. IEEE Internation Conference on Mobile and Ubiquitous Systems. (2004)
5. G. Judd and P. Steenkiste.: Providing Contextual Information to Ubiquitous Computing Applications. In Proc. IEEE Conference on Pervasive Computing and Communiations. (2003)
6. G. Rey and J. Coutaz.: The Contextor Infrasturcture for Context-aware Computing. In Proc. Workshop on the Component-oriented approach to context-aware systems (ECOOP04). (2004)
7. H. Lei, D. Sow, J. Davis, G. Banavar, and M. Ebling.: The Design and applications of a Context Service. Mobile Computing and Communications Review, Vol 6, Num. 4.
8. J. Hong and J.A. Landay.: An Infrastructure Approach to Context-Aware Computing, Human Computer Interaction, Vol. 16. (2001) 287-303
9. K. Koh, C. Choi, K. Park, S. Lim and S. Kim.: A Multilayered Context Engine for the smartHome. In Proc. International Conference on Computer Science, Software Engineering, Information Technology, E-business, and Applications. (2004)
10. M. Benerecetti, P. Pouquet, M. Bonifacio.: Distributed Context-Aware Systems. Human Computer Interaction. Vol. 16 (2001) 218-228.
11. M. Weiger.: The computer for the 21st Century. Scientific American, Vol. 265, No. 3. (1991) 66-75
12. O. Lehmann, M. Bauer, C. Becker, and D. Nicklas.: From Home to World – Supporting Context-aware Applications through World Models. In Proc. IEEE Conference on Pervasive Computing and Communications. (2004)
13. R. Glassey, G. Stevenson, M. Richmond, P. Nixon, S. Terzis, F. Wang, and I. Ferguson.: Towards a Middleware for Generalised Context Managerment. In Proc. MPAC (2003)
14. S. Jang and W. Woo.: Ubi-UCAM: A Unified Context-Aware Application Model. Lecture Notes in Artificial Intelligene, Vol. 2680. Spring-Verlag. (2003) 178-189.
15. T. Winograd.: Architectures For Context. Human Computing Interaction, Vol. 16. (2001) 401-419
16. W. N. Shilit.: A System Architecture for Context-Aware Mobile Computing. Ph.D. Thesis, Columbia University (1995)
17. X. Wang, J.S. Dong, C. Chin, S. Hettiarachchi, and D. Zhang.: Semantic Space: An infrastructure for Smart Spaces, Pervasive Computing IEEE, Vol 03, Issue 3. (2004)

Adaptive Voice Smoothing with Optimal Playback Delay Based on the ITU-T E-Model

Shyh-Fang Huang[1], Eric Hsiao-Kuang Wu[2], and Pao-Chi Chang[3]

[1] Department of Electronic Engineering, National Central University, Taiwan
hsf@vaplab.ee.ncu.edu.tw
[2] Department of Computer Science & Information Engineering,
National Central University, Taiwan
hsiao@csie.ncu.edu.tw
[3] Department of Communications Engineering,
National Central University, Taiwan
pcchang@ce.ncu.edu.tw

Abstract. Perceived voice quality is mainly affected by IP network impairments such as delay, jitter and packet loss. Adaptive smoothing buffer at the receiving end can compensate for the effects of jitter based on a tradeoff between delay and loss to archive a best voice quality. This work formulates an online loss model which incorporates buffer sizes and applies the ITU-T E-model approach to optimize the delay-loss problem. Distinct from the other optimal smoothers, the proposed optimal smoother suitable for most of codecs carries the lowest complexity. Since the adaptive smoothing scheme introduces variable playback delays, the buffer re-synchronization between the capture and the playback becomes essential. This work also presents a buffer re-synchronization algorithm based on silence skipping to prevent unacceptable increase in the buffer preloading delay and even buffer overflow. Simulation experiments validate that the proposed adaptive smoother archives significant improvement in the voice quality.

1 Introduction

The rapid progress of the development of IP-base network has enabled numerous applications that deliver not only traditional data but also multimedia information in real time. The next generation network, like an ALL-IP network, is a future trend to integrate all heterogeneous wired and wireless networks and provide seamless world-wide mobility. In an All-IP network, one revolution of the new generation Internet applications will realize VoIP services that people can talk freely around through the mobile-phones, the desktops and VoIP telephones at any time and place. Unfortunately, the IP-based networks do not guarantee the available bandwidth and assure the constant delay jitters (i.e., the delay variance) for real time applications. In other words, individual transmission delays for a given flow of packets in a network may be continuing to change caused by varying traffic load and differing routing paths due to congestions, so that the packet network delays for a continuous series of intervals (i.e. talkspursts) at the receiver may not be the same (i.e. constant) as the sender. In addi-

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 805 – 815, 2005.
© IFIP International Federation for Information Processing 2005

tion, a packet delay may introduce by the signal hand-out or the difference of bandwidth transportation in wireless/fixed networks.

For delay sensitive applications, a dominating portion of packet losses might be likely due to delay constraint. A late packet that arrives after a delay threshold determined by playback time is treated as a lost packet. A tight delay threshold not only degrades the quality of playback but also reduces the effective bandwidth because a large fraction of delivered packets are dropped. In fact, delay and loss are normally not independent of each other. In order to reduce the loss impact, a number of applications utilize an adaptive smoothing technique in which buffers are adopted to reduce the quality damage caused by loss packets. However, a large buffer will introduce excessive end-to-end delay and deteriorate the multimedia quality in interactive real-time applications. Therefore, a tradeoff is required between increased packet loss and buffer delay to achieve satisfactory results for playout buffer algorithms.

In the past, the works on the degradation of the voice quality consider the effect of packet loss, but not that of packet delay. Within literature on predicting delays, the use of Pareto distribution in [1] is of computing the distribution parameters and rebuilding the new distribution to predict the next packet delay, and the use of neural network models to learn traffic behaviors [2]. The use of Pareto distribution or a neural network model requires relatively high complexity or a long learning period. Therefore, we consider the smoothers [3]-[9] which employ statistical network parameters related with the voice characteristic, i.e. loss, delay and talk-spurt that have significant influence to the voice quality. They detect delay spike in traffic and quickly calculate the required buffer size to keep the quality as good as possible.

The E-model is a computational model, standardized by ITU-T in G.107, G.109 and G.113 which uses the various transmission parameters to predict the subjective quality of packetized voice. Unfortunately, the E-model is complex to analyze in the optimization process. An alternative study is to apply a simplified E-model, first proposed by R. Cole and J. Rosenbluth [10], based upon observed transport measurements in the VoIP gateways and the transport paths. Authors indicated the simplified E-model method requires more pattern cases for traces to enhance the validation. Atzori and Lobina [11], and L. Sun and E. Ifeachor [12] proposed to utilize a simplified E-model that considers the loss and delay together to set a dijitter time, which is the optimal playout delay derived by a dynamic programming-based solution. However, the usability and the accuracy of a simplified E-model will be limited by non-typical traffic patterns.

For perceptual-based buffer optimization schemes for VoIP, voice quality is evaluated as a key metric since it represents user's perceived QoS. However, it requires an efficient and accurate objective way to optimize perceived voice quality. To consider the well-defined delay and loss impairments of the E-model, we employ a complete E-model for the quality optimization to obtain optimal perceived voice quality.

In a packet switching network, without a resynchronization scheme, a playback clock with a minor frequency error will eventually cause a buffer overflow or an underflow at the receiving end. The overflow packets are usually discarded due to the finite buffer size and the real-time requirement. This discontinuity caused by discarded packets might create an unpleasant effect to the playback quality because the lost packets could be the important part of the signals. This effect is more serious for audio signals than video signals because human ears are more sensitive to the continuity of sounds than human eyes.

The contributions of this paper are three-fold: (i) A new method optimizing voice quality for VoIP is easily applied to codecs which were well-defined in the ITU-T E-model. (ii) Different from the other optimal smoothers, our optimal smoother has the lowest complexity with $O(n)$. (iii) A feasible scheme is introduced to solve the buffer re-synchronization problem.

2 Related Work

The Spike Detection (SD) algorithm has been studied by many researchers [3]-[9]. A delay spike is defined as a sudden and significant increase of network delay in a short period often less than one round-trip. This algorithm adjusts the smoothing size, i.e. playback delay, at the beginning of each talk-spurt. The results of this algorithm are therefore compared to the results obtained herein.

The SD Algorithm in [3] estimates the playout time p_i of the first packet in a talk-spurt from the mean network delay d_i and the variance v_i for packet i as

$$p_i = t_i + d_i + \gamma v_i \tag{1}$$

where t_i represents the time at which packet i is generated at the sending host and γ is a constant factor used to set the playout time to be "far enough" beyond the delay estimate such that only a small fraction of the arriving packets could be lost due to late arrival . The value of $\gamma = 4$ is used in simulations [3]. The estimates are re-computed each time a packet arrives, but only applied when a new talk-spurt is initiated.

The mean network delay d_i and variance v_i are calculated based on a linear recursive filter characterized by the factors α and β.

$$\begin{cases} If\ n_i > d_{i-1} \Rightarrow \begin{cases} d_i = \beta d_{i-1} + (1-\beta) n_i \\ v_i = \beta v_{i-1} + (1-\beta)\left|d_{i-1} - n_i\right| \end{cases} & (SPIKE_MODE) \\ If\ n_i \le d_{i-1} \Rightarrow \begin{cases} d_i = \alpha d_{i-1} + (1-\alpha) n_i \\ v_i = \alpha v_{i-1} + (1-\alpha)\left|d_{i-1} - n_i\right| \end{cases} \end{cases} \tag{2}$$

where n_i is the total "delay" introduced by the network [3] and typical values of α and β are 0.998002 and 0.75 [1], respectively.

The decision to select α and β is based on the current delay condition. The condition $n_i > d_{i-1}$ represents network congestion (*SPIKE_MODE*) and the weight β is used to emphasize the current network delay. On the other hand, $n_i \le d_{i-1}$ represents network traffic is stable, and α is used to emphasize the long-term average.

In estimating the delay and variance, the SD Algorithm utilizes only two values, α and β, which are simple but may not be adequate, particularly when the traffic is unstable. For example, an under-estimated problem is when a network becomes

spiked, but the delay n_i is just below the d_{i-1}, the SD Algorithm will evaluate the network to be stable and will not enter the *SIPKE_MODE.*

3 Adaptive Smoother with Optimal Delay-Loss Trade Off

The proposed optimal smoother is derived using the E-model to trade off the delay and loss. This method involves, first, building the traffic delay model and the loss model. Second, the delay and loss impairments of the E-model are calculated according to the delay and the loss models. Third, the E-model rank R is maximized and thus the delay and loss optimized solution is obtained.

3.1 E-Model Description

In the E-model, a rating factor R represents voice quality and considers relevant transmission parameters for the considered connection. It is defined in [13] as:

$$R = Ro - Is - Id - Ie_eff + A \tag{3}$$

Ro denotes the basic signal-to-noise ratio, which is derived from the sum of different noise sources which contain circuit noise and room noise, send and receive loudness ratings. *Is* denotes the sum of all impairments associated with the voice signal, which is derived from the incorrect loudness level, non-optimum sidetone and quantizing distortion. Id represents the impairments due to delay of voice signals, that is the sum of Talker Echo delay (*Idte*), Listener Echo delay (*Idle*) and end-to-end delay (*Idd*). Ie_eff denotes the equipment impairments, depending on the low bit rate codecs (*Ie, Bpl*) and packet loss (*Ppl*) levels. Finally, the advantage factor *A* is no relation to all other transmission parameters. The use of factor *A* in a specific application is left to the designer's decision.

3.2 The Delay and Loss Models in E-Model

For perceived buffer design, it is critical to understand the delay distribution modeling as it is directly related to buffer loss. The characteristics of packet transmission delay over Internet can be represented by statistical models which follow Exponential distribution for Internet packets (for an UDP traffic) has been shown to consistent with an Exponential distribution [14]. In order to derive an online loss model, the packet end-to-end delay is assumed as an exponential distribution with parameter $1/\mu$ at the receiving end for low complexity and easy implementation. The CDF of the delay distribution $F(t)$ can also be represented by [15]

$$F(t) = 1 - e^{\mu^{-1}t} \tag{4}$$

and the PDF of the delay distribution $f(t)$ is

$$f(t) = \frac{dF(t)}{dt} = \mu^{-1} e^{-\mu^{-1}t} \tag{5}$$

where μ is defined as the inverse of the average mean delay.

In a real-time application, a packet loss that is solely caused by extra delay can be derived from the delay model $f(t)$. The value of t_b represents the smoothing time for a smoother. When a packet delay exceeds t_b, a packet loss will occur. The loss function $l(t_b)$ can be derived as

$$l(t_b) = \int_{t_b}^{\infty} f(t)dt = \left(-e^{-\lambda t}\right)\Big|_{t_b}^{\infty} = -e^{-\infty} + e^{-\mu^{-1}t_b} = e^{-\mu^{-1}t_b} \tag{6}$$

3.3 Optimization on E-Model

The delay and loss factors over transmission have greater impacts to the voice quality than the environments or equipments. To simplify the optimization complexity, and investigate on delay and loss impairments, we make three assumptions in a communication connection as the following: (i). The circuit noise, room noise and terminate signals will not change. (*Ro* and *Is* are fixed). (ii). An echo delay in the Sender/Receiver will not change. (Idte and Idle are fixed). (iii). A codec will not change (*Ie* is fixed). In [13], *R* is rewritten as Eq. (7)

$$R = (Ro - Is - Idte - Idle + A) - Idd - Ie_eff \tag{7}$$

where *Idd* is approximated by

$$Idd = 25\left\{\left(1 + X^6\right)^{1/6} - 3\left(1 + \left[\frac{X}{3}\right]^6\right)^{1/6} + 2\right\},\ X = \frac{ln\left(\frac{t}{100}\right)}{ln(2)}, \tag{8}$$

when $T_a > 100$ ms and $Idd = 0$ when $T_a \le 100$,

and

$$Ie_eff = Ie + (95 - Ie)\cdot\frac{Ppl}{Ppl + Bpl} \tag{9}$$

Factors *Ie* and *Bpl* are defined in [16] and T_a is one-way absolute delay for echo-free connections.

Due to the three assumptions above, the optimization process can be concentrated on the parameters of *Idd* and *Ie_eff*. Eq. (7) is derived to yield Eq. (10)

$$R = Constant - 25\left\{\left(1 + X^6\right)^{1/6} - 3\left(1 + \left[\frac{X}{3}\right]^6\right)^{1/6} + 2\right\} - (95 - Ie)\cdot\frac{Ppl}{Ppl + Bpl}, \tag{10}$$

when $t > 100$ ms

The differential equation dR/dt is assigned to zero to maximize R to yield the best quality. According to Eq. (6), the loss probability $Ppl = e^{-\mu^{-1}t_b}$, so we can get

$$R' = 25 * \left\{ \left(1 + X^6\right)^{\frac{-5}{6}} X^5 X' - \left[1 + \left(\frac{X}{3}\right)^6 \right]^{\frac{-5}{6}} \left(\frac{X}{3}\right)^5 X' \right\} - \frac{\mu^{-1} e^{-\mu^{-1}t} \cdot Bpl}{Bpl^2 + e^{-2\mu^{-1}t} + 2 \cdot Bpl \cdot e^{-\mu^{-1}t}} = \quad (11)$$

$$, \; X = \frac{\log \frac{t}{100}}{\log 2}, \; X' = \frac{1}{\log 2 \cdot t}$$

The solutions for t are difficult to get directly from Eq. (11) since it contains the complex polynomial and exponential function, Therefore, we will try to solve the best smoothing time t with a numerical approach.

We notice the following three conditions. (i). In Eq. (8), when the smoothing time $t \le 100$ ms, *Idd* is zero (no delay impairment). It implies a smoother should set the minimum smoothing delay to 100 ms to prevent the most packet loss. (ii). The maximum end-to-end delay of 250ms is acceptable for most user applications to prevent serious voice quality destruction. (iii). For a common low bit rate codec, like G.723.1 and G.729, the frame rate is 30 ms and 10 ms, respectively, so the gcd(10,30) is 10 ms. Based on the above conditions, we can study the fifteen cases, $t_1 = 110\,ms$, $t_2 = 120\,ms$, …, $t_{15} = 250\,ms$, to calculate the correspondence, μ_1, μ_2, …, μ_{15}, by the numerical analysis in Eq. (11) and an error is less than 0.001. Table 1 shows the smoothing time t corresponding to μ. We can observe that as μ increases, the smoother will enlarge the smoothing time to smooth the late packets. According to Table 1, the proposed smoother will calculate the current μ ($\mu_{current}$) at the beginning of each talk-spurt and search for a minimum n which satisfies $\mu_n \ge \mu_{current}$. The optimal smoothing time will be $100 + n * 10$ ms to keep the optimal voice quality.

Table 1. The relation of smoothing time and arrival rate

smoothing time	μ (1/sec)	smoothing time	μ (1/sec)
$t_1 = 110\,ms$	$\mu_1 = 9.71$	$t_9 = 190\,ms$	$\mu_9 = 16.95$
$t_2 = 120\,ms$	$\mu_2 = 10.64$	$t_{10} = 200\,ms$	$\mu_{10} = 17.86$
$t_3 = 130\,ms$	$\mu_3 = 11.49$	$t_{11} = 210\,ms$	$\mu_{11} = 18.52$
$t_4 = 140\,ms$	$\mu_4 = 12.35$	$t_{12} = 220\,ms$	$\mu_{12} = 19.61$
$t_5 = 150\,ms$	$\mu_5 = 13.33$	$t_{13} = 230\,ms$	$\mu_{13} = 20.41$
$t_6 = 160\,ms$	$\mu_6 = 14.08$	$t_{14} = 240\,ms$	$\mu_{14} = 21.28$
$t_7 = 170\,ms$	$\mu_7 = 14.93$	$t_{15} = 250\,ms$	$\mu_{15} = 22.22$
$t_8 = 180\,ms$	$\mu_8 = 15.87$		

4 Buffer Re-synchronization

A necessary condition that a smoother can work correctly is the synchronization between the capture and the playback. This section proposes a buffer re-synchronization machine (BRM) to help synchronization and the clock drift analysis of re-synchronization to validate the effectiveness.

4.1 Buffer Re-synchronization Machine

This work proposes a synchronization scheme that segments audio signals by detecting silences. The mismatch between the capture and the playback clocks is solved by skipping silences at the receiving end. The duration of the silent period may be shortened negligibly degrading the quality of playback. An active packet contains voice-compressed data, whereas a silent packet does not. Skipping some silent packets will not significantly degrade the quality of the voice, but can efficiently prevent the buffer from overflowing. Notably, k (could be adjusted) continuous silent packets could be utilized to separate different talkspurts.

Figure 1 depicts the buffer re-synchronization algorithm. Init-state, Smooth-state, Play-state and Skip-state are used to represent the voice conference initialing, the buffer smoothing, the buffer playing out, and the silent packets skipping, respectively, and "A" and "S" represents an active packet and a silent packet, respectively.

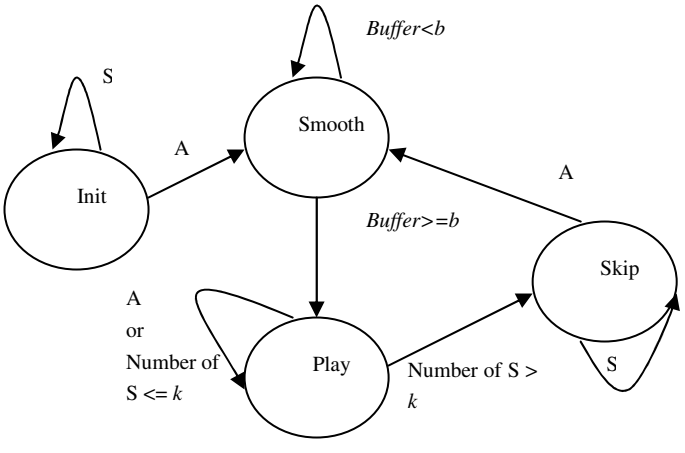


Fig. 1. Buffer Re-synchronization Machine

In the Init-state the buffer waits for the first arriving packets to initialize a voice conference. If Init-state receives an "S", it stays in Init-state; otherwise when an "A" is received, the Smooth-state is activated to smooth the packets. In the Smooth-state, the smoothing time b is computed by applying the optimal adaptive smoother algorithm dynamically. When the buffer smoothing time is over b, the Play-state is

activated; otherwise it stays in Smooth-state for smoothing. In the Play-State the packet is fetched from the buffer and played out. In fetching process, when it encounters three consecutive S packets, implying that the talk-spurt can be ended, the buffer re-synchronization procedure then switches to the Skip-state. In the Skip-state, if "A" is fetched from buffer, it means the new talk-spurt has begun, and then it skips remained silent packets in the buffer, and switches to the Smooth-state to smooth the next talk-spurt. Otherwise, if "S" is fetched from buffer, it implies current talk-spurt is not ended and will be decoded to play out at the same state.

With the above four-state machine, the smoother can smooth the packets at the beginning of the talkspurt to avoid buffer underflow in the Smooth-state and skip the silent packets at the end of the talkspurt to prevent the overflow in the Skip-state.

4.2 Effectiveness of Re-synchronization

To demonstrate the effectiveness of re-synchronization machine for buffer overflow, we analyze the clock inconsistence constraint as the following. C_s and C_r represent the sender clock (frame/sec) and the receiver clock, respectively, and M_a and M_s denote the mean active packets and mean silent packets in a talkspurt, respectively. The buffer overflow caused by the clock inconsistence (difference) will occur when C_s is larger than C_r condition. $C_s - C_r$, the difference value by subtracting the receiver clock from the sender clock, represents the positive clock drift between the sender and the receiver. Therefore, $(C_s - C_r) * ((M_a + M_s) * frame_time)$ represents the mean extra buffer size caused by the positive clock drift for a mean talkspurt time. In order to distinguish the consecutive talkspurts, at lease k silent packets are utilized. Therefore, the smoother has $M_s - k$ silent packets to be skipped and resynchronizes with the following talkspurt. When the re-synchronization machine satisfies

$$(C_s - C_r) * ((M_a + M_s) * frame_time) \le (M_s - k), \tag{12}$$

the buffer overflow caused by the positive clock drift will not occur.

5 Simulation

5.1 Simulation Configuration

A set of simulation experiments are performed to evaluate the effectiveness of the proposed adaptive smoothing scheme. The OPNET simulation tools are adopted to trace the voice traffic transported between two different LANs for a VoIP environment. Ninety personal computers with G.729 traffics are deployed in each LAN. The duration and frequency of the connection time of the personal computers follow Exponential distributions. Ten five-minute simulations were run to probe the backbone network delay patterns, which were used to trace the adaptive smoothers and compare the effects of the original with the adapted voice quality latter.

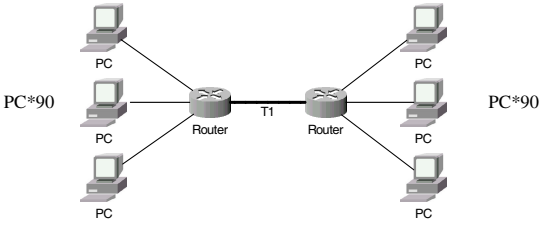
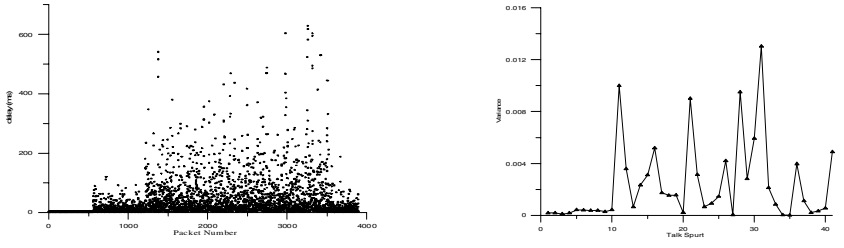


Fig. 2. The simulation environment of VoIP



(a) The delay of traffic (b) The variance of traffic

Fig. 3. VoIP traffic pattern

Fig. 2 shows the typical network topology in which a T1 (1.544 Mbps) backbone connects two LANs, and 100 Mbps lines are connected within each LAN. The propagation delay of all links is assumed to be a constant value and will be ignored (the derivative value will be zero) in the optimization process. The buffer size of the bottlenecked router is assumed to be infinite since the performance comparison of adaptive smoothers will be affected by overdue packet loss (over the deadline) and not affected by the packet loss in router buffer. The network end-to-end delay of a G.729 packet with data frame size (10 bytes) and RTP/UDP/IP headers (40 bytes) is measured for ten five-minute simulations by employing the OPNET simulation network. Figure 3(a) and 4(b) list one of the end-to-end traffic delay patterns and the corresponding delay variances for VoIP traffic observed at a given receiver.

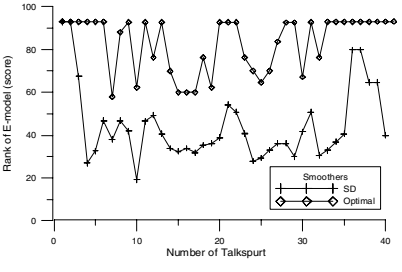


Fig. 4. The quality scores of smoothers

5.2 Voice Quality in Smoothers

The test sequence is sampled at 8 kHz, 23.44 seconds long, and includes English and Mandarin sentences spoken by male and female. Fig. 4 shows the E-model score R of the voice quality. It shows that the optimal method has the significant improvement in the voice quality over SD smoother, because our proposed optimal smoother truly optimizes with the delay and loss impairments in a transmission planning of the E-model.

5.3 Re-synchronization Effectiveness for the Positive Clock Drift

A listening evaluation experiment was performed to analyze the required proper number of silent packets to segment the consecutive talk-spurts well. It was found in our experiments that at least three silent packets (e.q. 10 ms per packet in G.729) are required to separate talkspurts.

We analyze the G.729 voice sources used in our experiments and find the percentage of the mean active and mean silent segment length in a talkspurt are 0.51 and 0.49 respectively, and the maximum talkspurt length is 257 packets. p =3 is adopted to segment the consecutive talkspurt. From the Eq. (12), we can calculate the effective clock drift between the sender and the receiver $C_s - C_r$ should be less than or equal to $(257 * 0.49 - 3) \Big/ \left((257) * 10 * 10^{-3} \right) = 47.8$ (frame/sec). Normally, the clock drift will not be over 47.8 (frame/sec) when a sender of G.729 transmits 100 (frame/sec) to the networks. Consequently, the smoother can avoid the buffer overflow well in our case.

6 Conclusion

This article proposes an adaptive smoothing algorithm that utilizes the complete E-model to optimize the smoothing size to obtain the best voice quality. The buffer re-synchronization algorithm is also proposed to prevent buffer overflow by skipping some silent packets of the tail of talk-spurts. It can efficiently solve the mismatch between the capture and the playback clocks. Numerical results have shown that our proposed method can get significant improvements in the voice quality which balances the target delay and loss.

References

[1] Brazauskas V., Serfling R.: Robust and efficient estimation of the tail index of a one-parameter pareto distribution. North American Actuarial Journal available at http://www.utdallas.edu/~serfling. (2000)

[2] Tien P. L., Yuang M. C.: Intelligent voice smoother for silence-suppressed voice over internet. IEEE JSAC, Vol. 17, No. 1. (1999) 29-41

[3] Ramjee R., Kurise J., Towsley D., Schulzrinne H.: Adaptive playout mechanisms for packetized audio applications in wide-area networks. Proc. IEEE INFOCOM. (1994) 680-686

[4] Jeske D. R., Matragi W., Samadi B.: Adaptive play-out algorithms for voice packets. Proc. IEEE Conf. on Commun., Vol. 3. (2001) 775-779
[5] Pinto J., Christensen K. J.: An algorithm for playout of packet voice based on adaptive adjustment of talkspurt silence periods. Proc. IEEE Conf. on Local Computer Networks. (1999) 224-231
[6] Liang Y. J., Farber N., Girod B.,: Adaptive playout scheduling using time-scale modification in packet voice communications. Proc. IEEE Conf. on Acoustics, Speech, and Signal Processing, Vol. 3. (2001) 1445-1448
[7] Kansal A., Karandikar A.: Adaptive delay estimation for low jitter audio over Internet. IEEE GLOBECOM, Vol. 4. (2001) 2591-2595
[8] Anandakumar A. K., McCree A., Paksoy E.: An adaptive voice playout method for VOP applications. IEEE GLOBECOM, Vol. 3. (2001) 1637-1640
[9] DeLeon P., Sreenan C. J.: An Adaptive predictor for media playout buffering. Proc. IEEE Conf. on Acoustics, Speech, and Signal Processing, Vol. 6. (1999) 3097-3100
[10] Cole R., Rosenbluth J.: Voice over IP performance monitoring, Journal on Computer Commun. Review, Vol. 31. (2001)
[11] Atzori L., Lobina M.: Speech playout buffering based on a simplified version of the ITU-T E-model. IEEE Signal Processing Letters, Vol. 11 , Iss 3. (2004) 382-385
[12] Sun L., Ifeachor E.: New models for perceived voice quality prediction and their applications in playout buffer optimization for VoIP networks. Proc. ICC. (2004)
[13] ITU-T Recommendation G.107,: The E-model, a Computational Model for use in Transmission Planning. (2003)
[14] Bolot J. C.: Characterizing end-to-end packet delay and loss in the internet. Journal of High-Speed Networks, Vol. 2. (1993) 305-323.
[15] Fujimoto K., Ata S., Murata M.: Statistical Analysis of Packet Delays in the Internet and its Application to Playout Control for Streaming Applications. IEICE Trans. Commun., Vol. E84-B, No. 6. (2001) 1504-1512
[16] ITU-T SG12 D.106: Estimates of Ie and Bpl for a range of Codecs. (2003)

The Wearable Computer as a Personal Station

Jin Ho Yoo[1] and Sang Ho Lee[2]

[1] Wearable Computing Research Team, ETRI,
161 Gajeong-dong, Yuseong-gu, Daejeon, 305-350, Korea
yoojh@etri.re.kr
[2] Dept. of Computer Science, Chungbuk National University,
48 Gaesin-dong, Cheongju, 361-763, Korea
shlee@chungbuk.ac.kr

Abstract. This paper introduces a wearable computer as a wearable personal station. We propose and implement wristwatch-style wearable personal station and its I/O devices in this paper. Nowadays the progress in miniaturizing more powerful computer systems and the availability of various devices around wearable computing will bring this technology to the edge of a new quality. Wearable computing is starting to become a product by itself. The function components of our wristwatch-style wearable computer include watch, PDA functions such as PIMS, address book, portable multimedia features, personal communication features and so on. We miniaturized these functionalities into our wearable small device. We also handle a study of USB implementation without wires for human interfaces. This paper describes a platform, hardware specifications and applications we have developed.

1 Introduction

Wearable computing becomes more and more feasible and receives growing attention throughout industry and the consumer marketplaces. Nowadays the progress in miniaturizing more powerful computer systems and the availability of various devices around wearable computing(e.g., wearable computers, high resolution Head Mounted Displays, interaction devices) will bring this technology to the edge of a new quality[4]. Wearable computing is starting to become a product by itself. Basic research is accompanied by an increasing amount of application and transfer related research.

This paper introduces our Wearable Personal Station named WPS as a wearable computer system. We propose and implement wristwatch-style wearable personal station in this paper. Because people generally keep watches on their wrists, watches are less likely to be misplaced compared to phones and pagers. For example a hip holster is not the best place to keep a cellular phone while sitting in a car and so people tend to keep them in the car seat and forget them when they leave the car in the parking lot[4].

Surely the watch form factor requires a relatively small screen size, and there is not much room for input devices or batteries. The value of a wristwatch platform depends on finding good solutions to these issues. To interact with the watch, we need both

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 816 – 825, 2005.
© IFIP International Federation for Information Processing 2005

hands since the hand on which the watch is worn is practically useless for controlling input devices on the watch. This wearable personal station plays a role as electronic secretary, network gateway. Wearable personal station is attached on human body and provides services.

2 System Overview

Wearable system is a computer system. So wearable system includes several hardware devices, software components, network devices and so on. Fig. 1 shows the system overview of our wearable personal station.

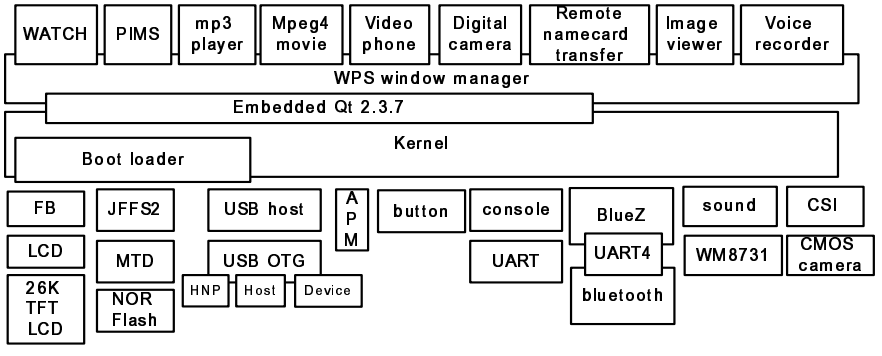


Fig. 1. The System Overview of Wearable Personal Station

3 Hardware Specification and an Outward Appearance

3.1 The Main Processor

We adopted DragonBall MX, the MC9328MX21(i.MX21) as our main processor[1]. The Dragon family of microprocessors has demonstrated leadership in the portable handheld market. Following on the success of the DragonBall MX(Media eXtensions) series, the MC9328MX21(i.MX21) provides a leap in performance with an ARM926EJ-S^{TM} microprocessor core that provides native security and accelerated java support in addition to highly integrated system functions. The i.MX21 processor features the advanced and power-efficient ARM926EJ-S^{TM} core operating at speeds up to 266MHz. We tried operating at speed up to 296 MHz. On-chip modules such as an MPEG-4 codec, LCD controller, USB OTG, CMOS sensor interface, and an AC97 host controller offer designers a rich suite of peripherals that can enhance any product seeking to provide a rich multimedia experience.

3.2 The Multimedia Modules and Display for the Wearable Personal Station

We use eMMA for multimedia function implementation in i.MX21. eMMA means the enhanced MultiMedia Accelerator which consists of the video Pre-processor(PrP), Encoder(ENC), Decoder(DEC) and Post-processor(PP). These blocks work together

to provide video acceleration and off-load the CPU from computation intensive tasks. While the encoder and decoder support only MPEG4-SVIP, the PrP and PP can be used for generic video pre and post processing such as scaling, resizing, and color space conversions.

The features of eMMA support private DMA between CMOS sensor interface module and pre-processor as data input and image scaling function. The eMMA encoder supports MPEG-4 and H.263, full conformance to ISO/IEC 14496-2 Visual Simple Profiles Levels 0 to 3, real-time encoding images of sizes from 32x32 up to CIF at 30 fps and camera stabilization. The eMMA decoder supports MPEG-4 and H.263, full conformance to ISO/IEC 14496-2 Visual Simple Profiles Levels 0 to 3, real-time encoding images of sizes from 32x32 up to CIF at 30 fps and camera stabilization. Post-processor receives the input data of YUV 4:2:0(IYUV, YV12) from system memory and does image resizing with upscaling ranging from 1:1 to 1:4 and downscaling ranging from 1:1 to 2:1 in fractional steps. This ratios provide scaling between QCIF, CIF, QVGA(320x240,240x320).

The Liquid Crystal Display Controller is embedded in the i.MX21 chipset. The Liquid Crystal Display Controller (LCDC) provides display data for external gray-scale or color LCD panels. The LCDC is capable of supporting black-and-white, gray-scale, passive-matrix color (passive color or CSTN), and active-matrix color (active color or TFT) LCD panels.

Camera module in the wearable personal station is controlled by CMOS sensor interface(CSI) characterized by CSI architecture, operation principles, and programming model. The CSI enables the i.MX21 to connect directly to external CMOS image sensors. CMOS image sensors are separated into two classes, dumb and smart. Dumb sensors are those that support only traditional sensor timing (Vertical SYNC and Horizontal SYNC) and output only Bayer and statistics data, while smart sensors support CCIR656 video decoder formats and perform additional processing of the image (for example, image compression, image pre-filtering, and various data output formats).

3.3 Multimedia Card/Secure Digital Host Controller

The MultiMediaCard (MMC) is a universal low cost data storage and communication media that is designed to cover a wide area of applications as electronic toys, organizers, PDAs and smart phones and so on. The MMC communication is based on an advanced 7 pin serial bus designed to operate in a low voltage range, at medium speed.

The Secure Digital Card (SD) is an evolution of MMC with an additional 2 pins as the same form factor and is specifically designed to meet the security, capacity, performance, and environment requirements inherent in newly emerging audio and video consumer electronic devices. An SD card can be categorized as SD Memory or SD I/O card, commonly known as SDIO. The SDIO card provides high-speed data I/O with low-power consumption for mobile electronic devices.

The Multimedia Card/Secure Digital Host module (MMC/SD) integrates MMC support with SD memory and I/O functions. We also mainly used MMC/SD as storage device. We newly made MMC/SD device driver for its use.

3.4 The Battery and an Outward Appearance

We have to get ultra slim battery for our wrist worn device. We need pocketing and free-stacking technology. This battery is lithium-ion rechargeable batteries tested as high temperature storage test (90 °C, 4hr), humidity test (60 °C, 90% RH, 1week), thermal shock test(-40 °C/60 °C, 10 cycles), safety test such as hot-box test (150 °C, 10min), nail penetration test, short circuit test, overcharge test (3C continuous over-charge) and long period storage test(25°C, 60°C, 80°C).

The outward appearance of our WPS hardware is as follows.

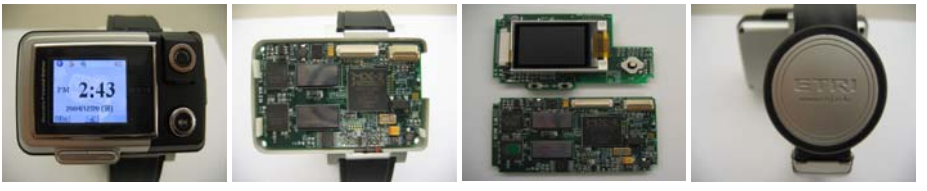


Fig. 2. Front Side, Front Inside, The PCBs, The Battery

4 System Software Specification

4.1 Operating System

We adopted linux operating system for free because open source. We patch linux-2.4.20 with mx2bsp patch. For building operating system, we needed to build a cross-development environment, including cross compilers, assemblers, and binary utilities that would let us generate code for the ARM architecture on our general purpose personal computer. Although there were many resources on the internet to help in this process, it took some time to get everything set up right. We found relevant pieces of source code and patches on the internet. We wrote some of the basic device drivers and modified the memory maps for our hardware configurations. In addition, we wrote some device drivers not supported directly. We used busybox and its init profile on jffs2 filesystem/mtd. We wrote newly mtd device driver for building jffs2 filesystem. We used NOR flash, 512 bytes erase block. We builded and used MMC/SD in evaluation phase. We mainly used MMC/SD, USB memory stick as the secondary memory device for transferring its applications and data. We adopted embedded Qt 2.3.7 as graphical user interface. We made WPS window manager based on embedded Qt 2.3.7. We also implement advanced power management.

4.2 Device Drivers

The CMOS camera is supported by CSI device driver. It supports for 352x288 or 640x480 pixel array, 30 frame/sec, CIF, QCIF, QQCIF or VGA, QVGA, enabling sub-sampling, still image capture.

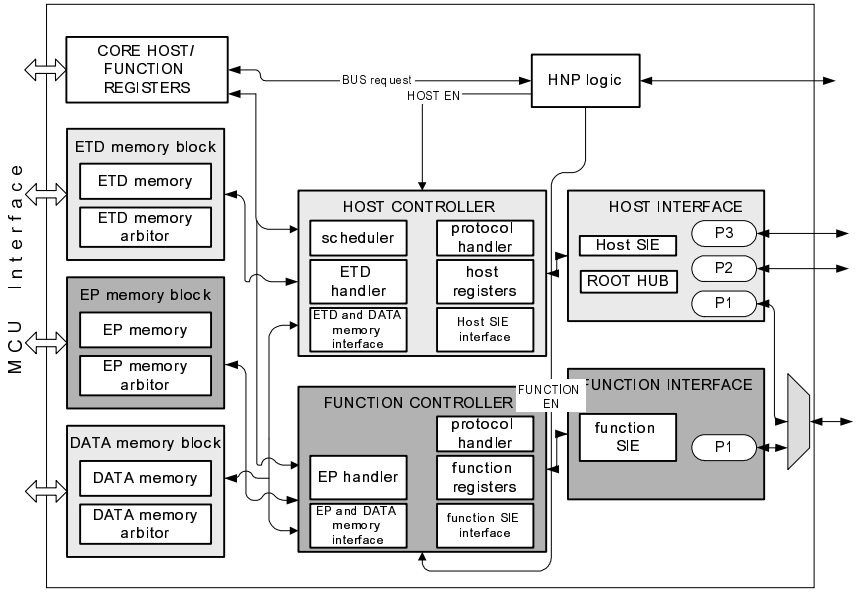


Fig. 3. The USB OTG Block Diagram

LCD driver supports 1.7 inch, 128x128 resolution TFT-LCD. We adopted 26K TFT-LCD and used frame buffer display. We wote MTD device driver newly for jffs2 filesystem. The flash device is interleaved by 2 and supports MTD. We used the flash chip 28F256 of intel.

The APM supports normal mode, doze mode and sleep mode with system software. The normal mode means the main processor and all devices are power on state. The doze mode means the main processor is power off state and all devices are power on state. The sleep mode means the main processor and all devices are power off state. This APM supports frequency scaling, not voltage scaling. The WPS consumes 113 mA in the normal mode, 85 mA in the doze mode, 35 mA in the sleep mode. We are trying to use frequency scaling for power scaling. The button is a simple device driver using interrupt service. We configure and make WM8731 device driver for supporting the sound.

Many of these portable devices would benefit from being able to communicate to each other over the USB interface, yet certain aspects of USB make this difficult to achieve. Specially, USB communication can only take place between a host and a peripheral. So OTG supplement to the USB is needed. The USB OTG block diagram is shown in Fig. 3[1]. We implement USBOTG device driver on the above blocks.

4.3 Making USB Wireless

USB continues to evolve as new technologies and products come to market. USB has already many application interfaces and support versatile device interfaces. It's already the de facto interconnect for PCs, and has proliferated into consumer electron-

ics(CE) and mobile devices as well. Like this, USB has built on many killer applications, many CE devices, many interfaces.

The growing use of wireless technology in PC, CE and mobile communications products, along with the convergence of product functionalities, calls for a common wireless interconnections. If our devices is unwired, it solves the tangle of wires. We want to use Legacy USB functionalities, portabilities, multimedia capabilities with wireless interconnection.

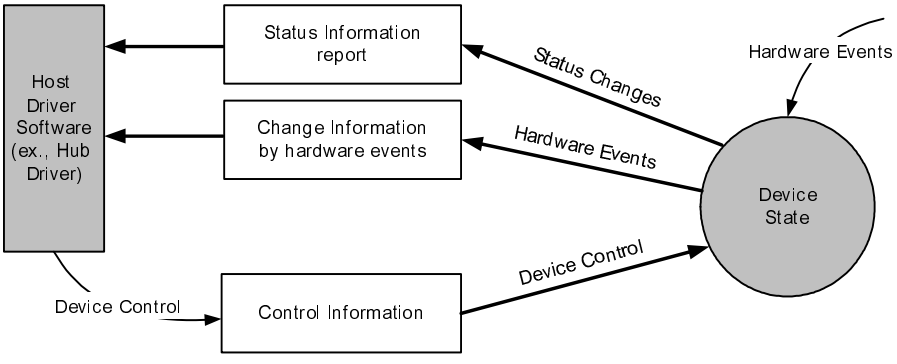


Fig. 4. Relationship of Status, Status Change, and Control Information to Device States

Making USB HUB Wireless. We will make the hub function of USB cordless, so it will connect host with devices without wires. The USB root hub is special device to be able to connect to another devices. We will make USB hub and USB host interface device wireless. There are many status report registers and events in hub. These are implemented by software modules and functions in the device driver level. And the memory uses and several data structures of USB host are also implemented by software modules and functions in the device driver level.

A hub is a standardized type of USB device. A hub is the only USB device that provides connection points for additional USB devices. The root hub is the hub that is an embedded part of every host controller. From an end user perspective, a hub provides the sockets that are used to plug in other USB devices. For making hub wireless, it needs to know the architecture requirements for the USB hub. It contains a description of the three principal sub-blocks: the hub repeater, the hub controller, and the transaction translator. There are major aspects of USB functionality that hubs must support: connectivity behavior, power management, device connect/disconnect detection, bus fault detection and recovery, high-speed, full-speed, and low-speed device support.

In case making USB wireless, it must support the above functionalities. Moreover, It needs the data structure and the resources for host functionalities, e.g. ETD, data memory.

Fig. 4 shows how status, status change, and control information relate to device states[2]. Hub or port status change bits can be set because of hardware or software events. When set, these bits remain set until cleared. While a change bit is set, the hub

continues to report a status change when polled until all change bits have been cleared by the USB system software.

The Convergence of IP Side and USB Side on Device Driver Level. This is software issues and the process of device emulation or device adaption. In here wireless media MAC/PHY may become 802.11a/b/g or UWB. We are going to make USB wireless for using legacy LAN and USB applications. When wireless USB device accesses wireless USB host, wireless USB host operates same as wire USB device plugs in wire USB host.

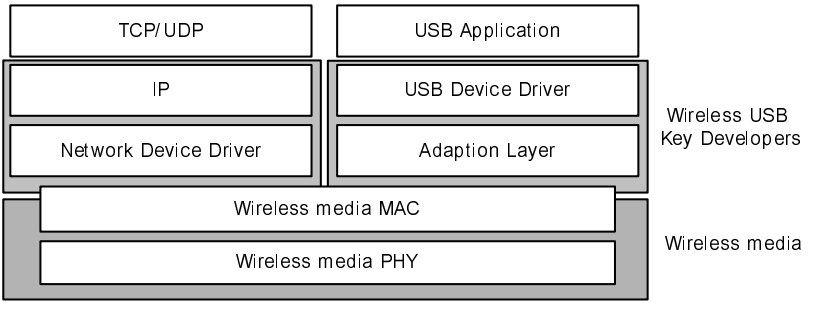


Fig. 5. The Stack of the Wireless Platform with IP

Fig. 5 shows the convergence of IP side and USB side. It will occurrs convergence on wireless media MAC level. The inputed packets should be divided into two parts, IP side and USB side. For that, USB packet should be defined as the special MAC packet. It recognizes whether IP side packet or USB device packet through checking the inputed packets in wireless media MAC device driver. The USB adaption layer has the data structure for USB hub and USB host functionalites. So USB device driver thinks USB adaption layer as USB real device. USB device driver drives USB adaption layer as the legacy USB device driver drives USB real device. USB adaption layer communicates the counterpart USB adaption layer of USB device through wireless media MAC/PHY transparently. Communication of wireless media MAC/PHY should meet timeliness and bandwidth of USB packet. For example, event, status report, interrupt, command for four mode, isochronous mode, bulk mode, interrupt mode, control mode. We should prepare software emulation functionalites for these implementations. This implementation provides the convenient and the effective on device usage with maintaining the legacy devices.

There are many candidates of wireless media. For example, UWB, 802.11a/b/g, Bluetooth. UWB among those candidates is high-powered because low-power, high bandwidth. UWB and the associated networking protocol efforts are in the early stages of development, and several key deployment scenarios are being defined and evaluated. UWB complements currently deployed wireless networks in the WLAN environment, plus it extends high bit-rate, multimedia connectivity to WPANs supporting PC and CE devices.

Many UWB components and systems are already in the testing and demonstration phases, with actual release dates for final consumer products expected in early 2005. We discussed software emulation for making USB wireless, However, This wireless

USB will be integrated into one chip solution not as software emulation. When wireless USB device accesses wireless USB host, wireless USB host operates same as wire USB device plugs in wire USB host.

4.4 Bluetooth Protocol Stack

We use the BlueZ bluetooth stack downloaded from internet[3]. The BlueZ has several device drivers, VHCI driver, UART driver, USB driver and so forth on the Bluetooth hardware. The BlueZ utilities and bluetooth applications use HCI sockets, L2CAP protocol, SCO sockets through the protocol interface of the BlueZ core driving VHCI driver, UART driver, USB driver and so on. The blueZ core also manages HCI life cycle and interfaces with the drivers of HCI UART, HCI USB, HCI PCMCIA. The blueZ manages the protocol of L2CAP, LMC/LC, RFCOMM, BNEP, CMTP, HIDP and SDP.

4.5 Videophone

The Wearable Personal Station has MPEG-4/H.263 codec. The video/audio encoding engine receives the MPEG-4 from MPEG-4/H.263 encoder pre-processing module and AAC from AAC encoder. This MPEG-4/H.263 encoder pre-processing module receives YUV from CSI and makes MPEG-4 stream. The video/audio decoding engine makes the AAC output for AAC decoder and makes MPEG-4 for MPEG-4/H.263 decoder post-processing module. LCDC receives RGB data from MPEG-4/H.263 decoder post-processing module and display on a screen.

4.6 Root Filesystem

The root file system is the jffs2 journalling file system on the MTD device for fast recovery. Actually the root file system is tested by system crash situation. The root file system includes / as a top directory, /bin as a directory for utility binary files, /sbin as a directory for system binary files, /dev as a directory for special device files, /drivers as a directory for dynamic kernel module files needed by kernel, /etc as a directory for configuration files, /lib as a directory for several library files, /mnt as a mount directory, /proc as virtual directory for system information, /tmp, /var as a directory for temporary files, /root as a root home directory, /usr as a system user directory. /usr/local as our special libraries and our special binaries.

5 Applications and User Interface

Our applications have watch function, PIMS, multimedia player, videophone, digital camera, namecard transfer and so on. Watch function includes time display, calendar display, alarm, time/date setting. PIMS means personal information management system which has address book, personal plan management, memo, electronic postit, electronic calculator and so on. Multimedia player plays mp3 files, MPEG-4 movie files. Videophone can do visual communication on WPS LCD display or HMD (Head

Fig. 6. MP3 Player, Digital Camera, Videophone, Namecard Transfer, MPEG-4 Player, Voice Recoder, Eval B'd with HMD, HMD output and Control Panel

Mounted Display, VGA). For these applications, we provide several intuitive menu screen displays for graphic user interface as shown in Fig. 6.

6 Conclusions

Personal computer(PC) has been evolved from desktop to portable PC such as tablet PC and PDA. Technology innovation on semiconductor have made it possible to package a reasonably powerful processor and memory subsystem with advanced input/output devices. Wearable computing becomes more and more feasible and receives growing attention throughout industry and the consumer marketplaces.

Our hardware technology enough to support many software applications and minimize its size. Our developing issue is a wearable personal station as a wearable personal platform with advanced I/O devices in the ubiquitous world. This wearable personal station will interoperate with the ubiquitous objects using network and I/O device.

Our work has convinced us that a lot of functionality can be packaged into small devices. We combine high-function miniature hardware with a robust operating system and well designed applications with intuitive user interfaces and so the wearable personal station forms the basis of personal station platform for the ubiquitous world.

References

1. Freescale,: i.MX21 application processor specification.
2. USB Implementers Forum, inc.,: On-The-Go Supplement to the USB 2.0 Specification.
3. BlueZ, http://www.bluez.org.
4. Chandra Narayanaswami, M. T. Raghunath,: Application design for a smart watch with a high resolution display, Fourth International Symposium on Wearable Computers, Oct 18-21, (2000)

Perception of Wearable Computers for Everyday Life by the General Public: Impact of Culture and Gender on Technology

Sébastien Duval and Hiromichi Hashizume

NII, Hitotsubashi 2-1-2, Chiyoda-ku, Tokyo 101-8430, Japan
duval@grad.nii.ac.jp, has@nii.ac.jp

Abstract. This paper examines the perception of wearable computers for everyday life by the general public, in order to foster the adoption of this technology. We present a social study that focuses on sensors, actuators, autonomy, uses, and privacy. Carried out in 2005, it considers gender and cultural disparities in two dissimilar groups: French (115 males, 59 females) and Japanese (61 males, 54 females) citizens. Acknowledging that exposition to wearables can alter perception about them, we designed a garment-shaped prototype to check our results, estimate shifts of perception, and define guidelines for equipment and services. We describe our prototype, and future experiments dealing with face-to-face contacts, community awareness, and relaxing environments.

1 Introduction

Current trends show that our environment is continuously being filled with new electronic systems. Although the technologies have matured, it remains unclear whether people will accept them willingfully and how we can facilitate their adoption. Wearables, for example, have been developped but their use remains marginal. To understand this situation and provide hints for related technologies, it is critical to examine social factors.

So far, research in wearable computing focused on experts and on technical aspects. We consider here the general public, its perception of wearables, needs, and requests. Because of the diversity of possible wearables, we cannot take people to laboratories and generalize their feedback. Therefore, we decided to begin with questionnaires. We oriented them on cyberclothes [4]: garments possessing special features that can be used as social markers or tools, and possess some autonomy. This was appropriate for laymen because garments are a natural part of their life, and because enhanced clothes are easy to imagine. Results were gathered in 2005, mainly in France and Japan, which allowed the identification of cultural convergences and divergences.

Studies on mobile phones show that perception of a device can vary after acquisition [11]. Although our questionnaire provides an *a-priori* perception of wearables, it provides limited hints for design, and does not reveal all pitfalls.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 826–835, 2005.
© IFIP International Federation for Information Processing 2005

As a consequence, we decided to also carry out experiments with prototypes focusing on problematic aspects revealed by the questionnaire. Feedback enabled us to check initial results, and define guidelines. This hybrid approach, combining the width of scope covered by questionnaires and the depth reached with experiments, favors reliable results.

Our paper is organized as follows. We present related works in section 2 then discuss our social study. In section 4 we describe the concept of cyberclothes. In section 5 we introduce our prototype, and planned experiments. In section 6 we discuss results and propose guidelines. Finally we conclude with future works.

2 Related Works

Diverse wearables were developed, providing functions of desktop computers (e.g. email, calendar, notes), or novel features such as help for navigation in urban spaces, recognition of persons encountered, and display of physiological data. These computers looked like boxes, accessories, or even garments. Digital accessories include jewelry [10], gloves that react to their wearer's stress [12], and badges displaying messages [2] [6]. Enhanced garments were mainly designed for experts [9] [13] [14]; *France Telecom*'s tee-shirt, which displays graphics, is a rare example of model dedicated to the general public.

The evaluation of several prototypes has led to the definition of metrics and guidelines for the design of wearables regarding comfort [1] [7] [8] and social interactions [14]. However, there are few studies on the perception and adoption of wearable computers. In 2001, a user study was planned to evaluate usability and acceptance of digital jewelry [10] but has not been carried out yet [1]. Studies of other devices, such as mobile phones, are of little use because their characteristics are so different: wearables can have any number and type of sensors, actuators, and can be used for a wide range of services.

3 Social Study About Perception of Wearables

The goal was to gather surface information about the general public's perception regarding wearable computers. Because people are not familiar with wearables, we focused on enhanced garments that respondents could easily imagine for uses in everyday life. This choice reduced the time required to answer questionnaires, and avoided confusion. We considered clear families of items (hardware, intelligence, information, uses), and everyday life situations. Therefore the questionnaire provides hints for wearables in general, and to a certain extent for other ubiquitous systems.

Our results reveal a significant effect of gender and culture on the perception of cyberclothes. Analyses show common interests (comfort, safety) and concerns (display of emotions) between the French and the Japanese. However they also indicate important divergences (e.g. system autonomy).

[1] E-mail exchanges with Christopher S. Campbell, IBM Almaden Research Center, 02-May-2005.

3.1 Method Followed and Scales

We designed the questionnaire with a pilot group, which led to rephrasing, and addition of an introduction. To enable quick answers, we limited the questionnaire to 2 pages with seven closed-ended questions and an open-ended one. For closed-ended questions, **participants rated assertions using a scale going from 1 to 5**: 1-strongly disagree, 2-disagree, 3-neither agree nor disagree, 4-agree, and 5-strongly agree. We considered that **a mean below 2.5 indicated a significant trend for rejection while a mean above 3.5 indicated acceptance**. Besides, we considered the existence of a significant gender effect when the difference between genders' mean values was superior to 0.5. The questionnaire was produced by native speakers in English, French, and Japanese.

Due to cultural and ecological specificities, answers can vary between populations. Therefore, we focused on two dissimilar cultures, while collecting other data worldwide for comparison purposes. A sample of 333 laymen (199 males, 134 females) ranging in ages from 9 to 67 replied in 2005. Respondents came from 25 countries, with two core groups (table 1) from France and Japan. First results for France were previously introduced in [3]. To reflect populations' heterogeneity, the questionnaires were provided in public places and over Internet. Respondents included artists, designers, librarians, reporters, students, teachers, researchers, engineers, secretaries, salesmen, managers, housewives, retirees, medical staff, soldiers, preachers, etc.

Table 1. Respondents (France and Japan)

	French Male	French Female	Japanese Male	Japanese Female
Number	115	59	61	54
Age Range	14 - 67	14 - 58	19 - 54	14 - 45
Age Mean (S.D.)	26 (09)	25 (09)	29 (08)	30 (07)

3.2 Perception of Wearables in France

Sensors and stimulators. As shown by scores between 3.7 and 4 (figure 1), French respondents are ready to accept garments that adapt their temperature to the environment, or analyze the air (smells, pollution, humidity, temperature). Results for recording graphics or sounds indicate a significant gender effect: French males are more attracted than females by the possibility to record graphics and sounds. Results for other features are not significant. Figure 2 indicates a high acceptance (3.6-4.4) of physiological monitoring to contact emergency services, evaluate sports performances, and adapt environments to users' needs. However, it shows an opposite trend for the disclosure of emotions (1.4-2.7), especially for women. There is a marked gender effect for physiological monitoring to adapt video games, females rejecting it whereas males remain neutral.

Attractive usages. Participants showed interest for three situations: in potentially dangerous situations (3.8-4.1), to communicate with disabled persons

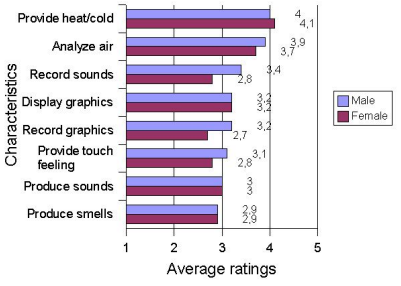


Fig. 1. Acceptance of perceptual sensing and stimulation (France)

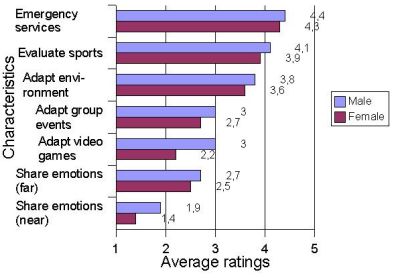


Fig. 2. Acceptance of physiological monitoring (France)

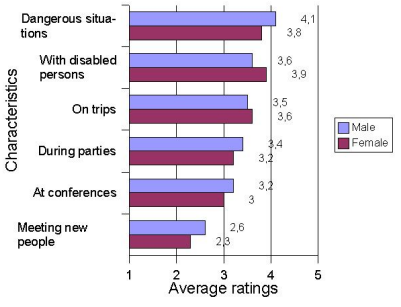


Fig. 3. Interest in usages (France)

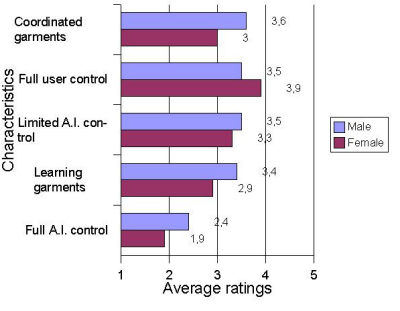


Fig. 4. Wishes for autonomy (France)

(3.6-3.9), or on trips (3.5-3.6). However figure 3 shows that enhanced garments are rejected for more common situations such as meeting new people. Besides, respondents rejected the ability to provide data about surrounding people (1.8-2.6). Getting information about ongoing tasks was not significantly appreciated (only 3.3). Comments expressed concerns about privacy.

Autonomy of the garments. Gender effect is strong, with males having a significant better acceptation of several features proposed. Respondents, especially women, indicated their will to own a system as much as possible under their direct control. This is shown on figure 4 by the rejection of full control by artificial agents (1.9-2.4), partial acceptance of limited artficial intelligence (3.3-3.5), and significant acceptance of full user control (3.5-3.9). The main reason indicated in comments highlighted a fear that the system might harm the wearer, with no possibility for her to fix the problem. Finally, respondents neglect the possibility for garments to learn from wearers' reactions (2.9-3.4) or to coordinate their actions (3.0 for women).

3.3 Perception of Wearables in Japan

Sensors and stimulators. Gender effect is significant for features related to perception. As shown on figure 5, respondents look forward (3.5-4.2) garments that

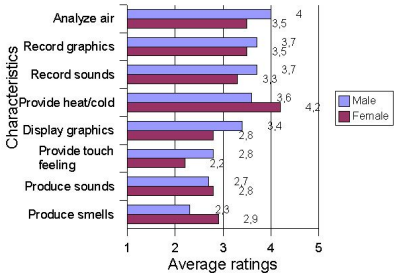


Fig. 5. Acceptance of perceptual sensing and stimulation (Japan)

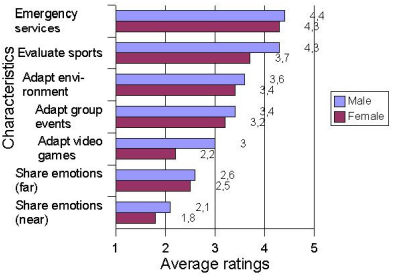


Fig. 6. Acceptance of physiological monitoring (Japan)

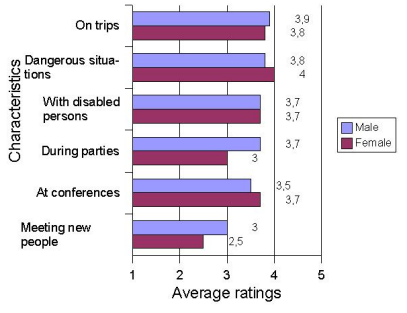


Fig. 7. Interest in usages (Japan)

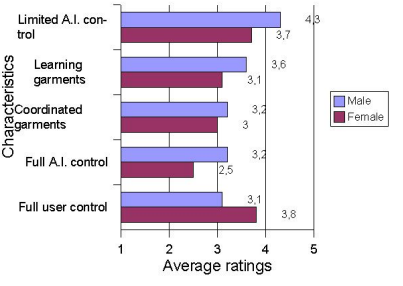


Fig. 8. Wishes for autonomy (Japan)

adapt their temperature to the environment, analyze the air, or record graphics. Males also consider acceptable to record sounds and display graphics on clothes; they reject the production of smells while females reject the generation of touch feelings. Figure 6 indicates a high acceptance (3.7-4.3) of physiological monitoring for emergencies and sports. Males are interested in the adaptation of their environment to fit their needs (3.6). However, there is an opposite trend for the disclosure of emotions (1.8-2.6). Females highly reject (2.2) physiological monitoring to adapt video games, while males remain neutral (3.0).

Attractive usages. Figure 7 shows that participants are attracted by proposed uses, except to meet new people (2.5-3.0). There is a gender effect for *meeting new people* and *parties*; female reject them strongly. Respondents reject the provision of data to surrounding people (2.5) but not for ongoing tasks (2.7-3.4).

Autonomy of the garments. As shown on figure 8, women reject full control of enhanced garments by artificial agents (2.4). Partial or full user control appears more attractive, women preferring full control (3.8), and male limited A.I. control (4.3). These aspects indicate a strong gender effect. The capacity to learn from wearers' reactions is attractive to males (3.6). Results for coordinating actions are not significant.

4 Wearables for the General Public: A Humane Philosophy

Focus on technics limited the development of wearables dedicated to the general public. With *cyberclothes*, we hope to replace human beings at the locus of attention. Creating this concept clarifies our goals, provides an alternative for the design and adoption of wearables on a wide-scale.

We define cyberclothes —in a nutshell— as garments that have special features improving a person's well-being, awareness, and sociability. This definition takes its roots in Maslow's hierarchy of human needs, usually represented as a pyramid, displaying the most fundamental needs at the bottom. Its five main levels are physiological, safety, belonging, esteem, and self-actualization needs. Based on this theory, a wearable that fulfills a person's needs should sustain in priority physiological needs, safety needs, and belonging needs.

Physiological needs include food, water, air, and sleep. Safety needs include physical and psychological security (e.g. health, comfort, absence of danger, good familial situation). Belonging needs are related to relationships with people at large, in clubs, at the office, etc. Informal interviews we carried out in France and Japan indicate an interest for wearables that can monitor the evolution of the body condition: calories absorbed and used, sleep duration, etc. The study presented previously also indicates an interest for wearables that increase comfort and safety. These results indicate that the orientations chosen for cyberclothes are pertinent.

Cyberclothes improve a person's well-being, which encompasses physiological and safety needs. This can be achieved with garments that monitor physiology and contact emergency services, provide guidance, adapt to the environment or adapt the environment to the user (temperature, light). They can also maintain peace of mind by conveying comforting news (kids, lover).

Cyberclothes improve a person's awareness, which help take good decisions and feel secure. Using sensors and Internet, cyberclothes can provide information about the user herself (movements, biosignals), other people (availability of coworkers), or the environment (pollution level).

Cyberclothes improve a person's sociability, sustaining mainly belonging needs. With actuators, garments can become a medium of communication. For example garment could change their shape, modify graphics on their surface, produce smells, process speech, compare profiles of wearers, etc. Uses include easing communication with deaf persons and finding people with similar interests.

5 Experiments Proposed to Check Our First Findings

To complement our social study, we also experiment with prototypes. Our applications deal with *face-to-face contacts*, *community awareness*, and *relaxing*

environments. We chose these to investigate privacy, relations with the environment, and full control by artificial agents, as well as the apparent rejection of emotion sharing and support for first contacts.

5.1 Prototype to Investigate Belonging and Safety Needs

We developed a prototype (figure 9) consisting of a jacket that processes data, accesses wireless networks, displays graphics, renders sounds, and acquires biosignals (skin conductivity, heartbeats). The two displays were placed to support conversations, and inform passersby about the wearer. The biosignals equipment aims at identifying the body condition, stress, calm, and intense emotions.

Fig. 9. Support for social events

5.2 Experiments: Contacts, Awareness, and Relaxation

Face-to-face contacts. Using JAVA and XML, we developed a system in which prototypes dynamically generate and display personalized slideshows of photos corresponding to common interests of surrounding wearers. Based on [5], we plan to check the interest of the service, evaluate controls for the system, and investigate private displays to inform wearers about their displays.

Community awareness. Belonging to a community implies keeping in touch and strengthening links with other members, which can be difficult due to temporal or spatial constraints. Our prototype can convey information and represent it with a *night sky* metaphor: it displays a sky where each star represent a community member, colors and intensities representing feelings (based on biosignals or manual choices). We are particularly interested in reactions of users regarding privacy and disclosure of emotions.

Relaxing environments. To improve people's life, we propose to use garments to generate healing, relaxing environments. We still have to develop this system, but we plan to use biosignals as input, to control music, light, and scents

around the wearer. This should help us both evaluate environments and generate relaxing ones.

6 Results and Guidelines

The results of our social study in France and Japan show common interests and concerns regarding cyberclothes, in two dissimilar cultures. Using these as a basis, we established guidelines to design wearables. Besides, the creation of our prototype and preparation of experiments raised social and technical issues.

6.1 Results: Cultural Convergences and Divergences

French and Japanese people show similar perceptions of wearable computers concerning the hardware and services. However, there are important differences about the control of the system, especially concerning the amount of artificial intelligence considered acceptable.

Wearables related to comfort and safety are perceived very positively; several respondents even proposed additional applications related to health: sleep monitoring, stress feedback, and support for diets. The French and Japanese give high ratings (3.5-4.2) to garments that analyze the air (e.g. pollution) and adapt their temperature to the environment. In addition, physiological monitoring is considered positively (3.4-4.4) to adapt the environment to users' needs, evaluate sportive performances, and inform emergency services. Finally, respondents had **interest in wearables for situations involving safety and fundamental communication abilities**: on trips (3.5-3.9), to communicate with disabled persons (3.6-3.9), or in potentially dangerous situations (3.8-4.1). There is little variation here between males and females.

However, the **perception of wearables for standard social uses is negative**, and results highlight a gender effect. French and Japanese respondents strongly reject the use of physiological monitoring to reveal emotions to distant or surrounding people (1.4-2.7). Comments indicate that respondents did not want to disclose their emotions because they considered it as private information, and potential harmful, especially if revealed in real-time. Besides both consider that support for meeting new people is not interesting (2.3-2.6). Females have significantly more negative views than males on these issues. In addition, respondents reject the ability to provide data about individuals present at a gathering (1.8-2.6). Finally, French people also reject the possibility to get information about the group as a whole whereas Japanese keep a neutral stance on this issue.

Cultural differences are most visible when considering the control of wearables. The preferred choice for the Japanese would be to use limited artificial intelligence (3.7-4.3) whereas French people prefer full user-control (3.5-3.9). In both cultures, full control by an artificial agent is seen very negatively by women (1.9-2.5); males have a more open attitude but it remains negative in France (2.4) and neutral in Japan (3.2). As highlighted during interviews,

the French and the Japanese are both afraid that mistakes of an artificial agent might harm them socially or physically, with no way to fix the problem quickly.

These results confirm that focusing on human fundamental needs is a good approach to foster the adoption of wearable computers. However they also indicate a significant gender effect, and possible difficulties raised by the presence of artificial agents to control the equipment.

6.2 Universal Guidelines for the Adoption of Wearables

Because French and Japanese cultures are very different, the similar trends revealed point to guides valid worldwide:

1 - Wearables should improve the comfort and safety of their wearer, and possibly of surrounding or distant people.
2 - Wearables should be able to communicate with other devices, and to suggest them a behavior based on knowledge about their wearer.
3 - Support for communication should focus on disrupted settings (e.g. on trips, or with disabled persons) rather than on standard situations.
4 - Design should be gender-oriented, taking into accounts the specific concerns of males and females.
5 - Full control of the system by an artificial agent should be avoided, and the autonomy and intelligence of the system should be selected based on cultural preferences.

6.3 Issues Raised

Our results raised two main issues: control of wearables, and privacy. The experiments planned raise additional questions about services specificities.

People feel unease with artificial intelligence. Respondents rejected full control of wearables by artififical agents. Interviews and comments gathered with the questionnaire show that people are afraid that wearables would harm them physically or socially. Examples include selecting a wrong temperature for the heating system, or displaying unappropriate pictures at the office.

Requirements for privacy can hamper services. The use of personal information (e.g. emotions, condition, and location) is the subject of numerous research and needs to be investigated further before reliable guidelines can be produced.

Each service will raise new problems. Wearables allow the creation of diverse services and designs. Therefore various issues arise. For example, the display of photos reflecting common interests requires content and context annotations. Other problems include asymmetry of services, displayers unaware of what viewers see on them, or people ignoring the meaning of public display metaphors.

7 Conclusions and Future Works

In this paper we studied the perception of wearable computers for everyday life by the general public. Our hybrid approach combines social studies and experiments with prototypes. We found common interests (comfort, safety) and fears

(control by an artificial agent) in France and Japan, revealed a gender-effect, and highlighted cultural differences. We introduced the concept of cyberclothes, a prototype, and future experiments. Finally we proposed design guidelines for the adoption of wearables, and raised related issues. From now on, we will carry our experiments to validate further our results, and acquire more insights regarding the design of wearable and ubiquitous systems.

Acknowledgment. The authors would like to thank Ryo Yoshinaka (University of Tokyo) and Takeshi Ozawa (University of Tokyo), for translating the questionnaire into Japanese.

References

1. Bodine, K., Gemperle, F. : Effects of functionality on perceived comfort of wearables. International Symposium on Wearable Computers. (2003) 57–61
2. Borovoy, R., Martin, F., Vemuri, S., Resnick, M., Silverman, B., Hancock, C., : Meme tags and community mirrors: moving from conferences to collaboration. Computer Supported Cooperative Work. (1998) 159–168
3. Duval, S., Hashizume, H. : Acceptance and Expectations for Cyberclothes by the General Public in France. Virtual Reality International Conference. (2005) 187–190
4. Duval, S., Hashizume, H. : Cyberclothes: personal media in everyday life. International Conference on Active Media Technology. (2005) 44–47
5. Duval, S., Hashizume, H. : Facilitating Face-to-Face First Contacts with Cyberclothes. Virtual Reality Society of Japan Annual Conference. (2004) 87–90
6. Falk, J., Bjork, S. : The BubbleBadge: a wearable public display. Computer-Human Interaction. (1999) 318–319
7. Gemperle, F., Kasabach, C., Stivoric, J., Bauer, M., Martin, R. : Design for Wearability. International Symposium on Wearable Computers. (1998) 116–122
8. Knight, J., Baber, C., Schwirtz, A., Bristow, H. : The comfort assessment of wearable computers. International Symposium on Wearable Computers. (2002) 65–72
9. Lind, E., Eisler, R., Jayaraman, S., Mckee, T. : A sensate liner for personnel monitoring applications. International Symposium on Wearable Computers. (1997) 98–105
10. Miner, C., Chan, D., Campbell, C. : Digital jewelry: wearable technology for everyday life. Computer-Human Interaction. (2001) 45–46
11. Palen, L., Salzman, M., Youngs, E. : Discovery and integration of mobile communications in everyday life. Personal and Ubiquitous Computing **5** (2001) 109–122
12. Picard, R., Scheirer, J. : The galvactivator: a glove that senses and communicates skin conductivity. International Conference on Human-Computer Interaction (2001)
13. Rantanen, J., Alfthan, N., Impio, J., Karinsalo, T., Malmivaara, M., Matala, R., Makinen, M., Reho, A., Talvenmaa, P., Tasanen, M., Vanhala, J. : Smart clothing for the arctic environment. ISWC (2000) 15–24
14. Toney, A., Mulley, B., Thomas, B., Piekarski, W. : Social weight: designing to minimise the social consequences arising from technology use by the mobile professional. Personal and Ubiquitous Computing (2003) 309–320

Videoconference System by Using Dynamic Adaptive Architecture for Self-adaptation

Chulho Jung, Sanghee Lee, and Eunseok Lee

School of Information and Communication Engineering Sungkyunkwan University,
300 Chunchun Jangahn Suwon, 400-746, Korea
{jesus98, neomine}@selab.skku.ac.kr
eslee@ece.skku.ac.kr

Abstract. According to the improvements in wireless Internet technology, multimedia applications create new challenges, which must be solved. Internet-based videoconferencing systems have many variable factors, such as changes in the system operating environment or operating status of the hardware, according to the operator using the system. In this paper, we propose the Videoconferencing System, with multi-agents, for efficient videoconferencing, which is able to adapt itself to these various factors. This videoconferencing system has a dynamic architecture which is able to change the own architecture in runtime.

1 Introduction

The improvements in wireless internet technology, create new challenges which must be solved. Internet based applications are facing various factors which affect Quality of Service, the dynamically changing network environment, diverse range of user devices and preferences. Multimedia applications, especially applications in end-to-end wireless networks, need to seriously consider the dynamic network environment and system resources. Internet videoconferencing systems are no exception to this rule, and need various structures and languages for their interoperation with various network systems and operating environments, which have resulted in the development of many new technologies [1], such as Self-Adaptive software. The Self-Adaptive software [1] uses technology that is able to understand, monitor and correct changes by program itself; therefore, it should already have data relating to the program needs, the know how to enable evaluation of these data and to the ability to respond to any changes. This Self-Adaptive software technology helps clients adapt to the videoconferencing environment. For this reason, Self-Adaptive software research has been applied to many Videoconferencing Systems. [2][6]

In this paper, we propose a videoconferencing system in which the clients are able to adapt to the operator's processing capability and network bandwidth. Also, our system includes specific functions that allow it to adapt to various network situations, operating systems (O.S.'s) and devices. Moreover, the system helps a client communicates with the other clients with a different operating platform. However, it is difficult for a mobile device to adapt itself to the videoconferencing environment, as the adaptation mechanism is optimized for a Desktop PC. Thus, we have designed dy-

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 836 – 845, 2005.
© IFIP International Federation for Information Processing 2005

namic adaptive architecture that changes the adaptation architecture of the devices according to the circumstances. For this, we have designed and implemented a system with an Architecture Manager, which decides and manages the client's architecture according to the particular environment.

Section 2 contains a summary of related works, in section 3 we describe the proposed system and section 4 discusses the implementation and evaluation of such software. Section 5 is a presentation of our conclusions.

2 Related Works

2.1 Dynamic Software Architecture

2.1.1 C2 – A Component- and Message-Based Architectural Style for GUI Software [7]

C2, which was developed at UCI(University of California - Irvine), is an architecture used to support application software development, which uses a component and message based format. A C2 system is composed of a hierarchy of concurrent components interlinked by connectors – message-routing devices – such that each component within the hierarchy can only be aware of those components "above", and completely unaware of components residing at the same level or "beneath". [7] During runtime, C2 can add, delete or rearrange components, and has been optimized for flexible components.

2.1.2 Weave – Using Weaves for Software Construction and Analysis [8]

Weaves, made by The Aerospace Corporation, are networks of concurrently executing tool fragments that communicate by passing objects, and have a dynamic, object-flow-centric architecture, designed for applications that are characterized by continuous or intermittent voluminous data flows and real-time deadlines.

Weaves embrace a set of architectural principles, known as the laws of blind communication: [8]

- No component in a network is able to recognize the sources of its input objects or the destinations of its output objects;
- No network component is able to recognize the semantics of the connectors that delivered its input objects or transmitted its output objects; and
- No network component is able to recognize the loss of a connection.

Weaves support component manipulation of a like form, emphasize the dynamic distribution, modification and rearrangement of connectors, and has been optimized for flexible connectors.

2.2 Video Conferencing System

2.2.1 JQOS – A QoS-Based Internet Videoconferencing System [2]

The JQOS system was developed at MCR Laboratory, Ottawa Univ. Canada, an is based on QoS; therefore, a QoS adaptation function is included in the Video Conferencing System.

It's functions are as follows:

- First, for the End-User, the Active QoS Self-Adaptive function is now based on the network transportation capability.
- Second, for the Receiver, the smart processing of the requirements from the receiver, and the measurement for proper adaptation of the current system QoS.
- Third, for the Receiver, an expression function of the receiver's interests relating to the transporting stream is required for the QoS adaptation.

The JQOS system limits the Self-Adaptive element to stream the quality of the user, as mentioned above. This system also uses RTPR (Real-time Transport Protocol Report) as a control method, from the RTP (Real-time Transport Protocol) of JMF[4], so the Self-Adaptive function in the JQOS system is only weak.

3 The Proposed System

3.1 Architecture-Based and Model-Based Adaptation

Our self-adaptive videoconferencing system is an "Architecture based approach to Self-Adaptive software"[3]. In this paper, our proposed system is reflected in the above self-adaptive software process. Also, we developed the Videoconferencing system, which is operated by activity, as follows in Fig.1.

For the adaptation to a dynamic environment, all environment information, such as the system components and theirs interactive environmental information, and the available resources and operating environments, are captured and modeled with respect to several multi-aspects. The term "multi-aspect" refers to four main aspects - an architectural aspect, a behavioral aspect, a resource aspect and an environment aspect. Each aspect is defined as follow:

The architectural aspect is used to describe configurative entities, such as the software architecture, its configurative elements, each element's operation or role, and the relationship of each element. Each element is represented by a node, referred to as a "Component", with the relationship of each component represented by arcs, which are referred to as 'connectors'. The behavioral aspect defines the 'Interactive' operation end events between components.

The resource aspect defines the general system environmental entities that execute the necessary videoconferencing system. The target of a system for modeling the resource aspect is classified into two categories, the static and dynamic contexts. The static contexts are elements that have less dynamic changing characteristics, such as the type of user device, OS, executing software and Player information. The dynamic contexts are dynamic changeable elements, such as the current CPU and memory usage, state of camera and audio devices, network bandwidth and user preference. The Environment aspect defines the external environmental element that is mutually applied or affected by the surroundings of the videoconferencing system.

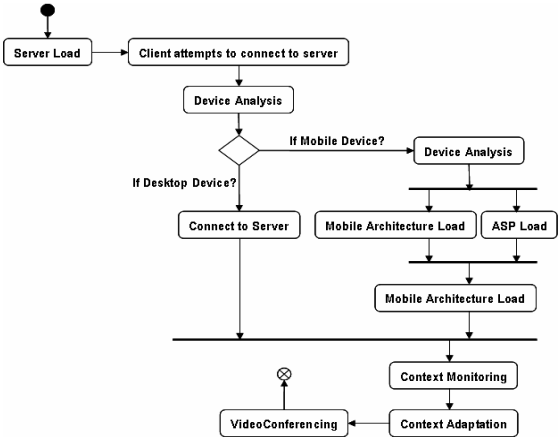


Fig. 1. Activity diagram of Our Videoconferencing System

These four aspects of modeling are graphically and texturally represented using the GME (Generic Modeling Environment) Tool. To adapt for executing system, Modeled elements must be transmitted to be realized by software, so we applied CC/PP technology to recognize system. Fig.2 shows the CC/PP format for recognize system.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:prf="http://www.testProfile.com/profiles/ccppschema-20030226#">
      <rdf:Description rdf:ID="MyProfile">
            <prf:component>
                  <rdf:Description rdf:ID="HardwarePlatform">
                        <prf:ScreenSize>84x30</prf:ScreenSize>
                        <prf:Type>PDA</prf:Type>
                        <prf:CPU>PPC</prf:CPU>
                        <prf:Memory>16MB</prf:Memory>
                  </rdf:Description>
            </prf:component>
      </rdf:Description>
</rdf:RDF>
```

Fig. 2. CC/PP format for recognize system

3.2 Overall Architecture

Our general approach for supporting the Dynamic Adaptation Architecture consists of several parts: the *Server, Client, Mobile-Client and Adaptation Proxy Server*. Fig.3. show the overall architecture.

The detailed architecture and algorithm of each part can be described as follows:

a) Server

The server is a manager, which manages the session, Adaptive Proxy Server and context, and consists of 1) Context DB, 2) Adaptive Proxy Server Manager (APSM), 3) Session Manager, 4) Context Manager and 5) Device Analyzer. The server is loaded into Desktop Pc.

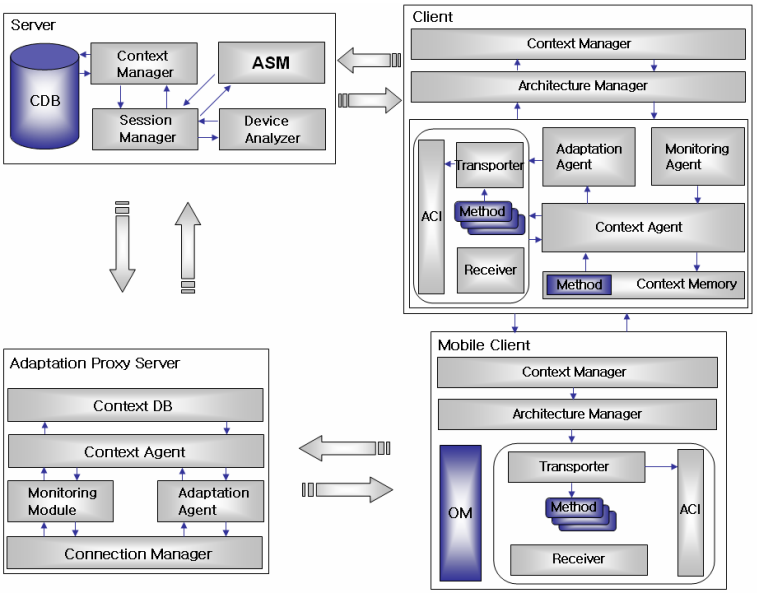


Fig. 3. Overall architecture

The Context DB is the stored context, by the Context Manager, of the connected client. The APSM manages the Adaptive Proxy Server. The functions of the APSM are as follows: 1) creates new APS; 2) assigns clients to APS; 3) controls client's number per APS; and 4) manages a session of executing APS. The policy of an APS creation mechanism is as follows:

1. *If there is no device with enough resource on the network, the APSM does not create an APS.*
2. *If there are devices with enough resources on the network, the APSM creates an APS.*
3. *If the number of clients per APS is larger than 15, the APSM creates a new APS. We investigate the relationships between the APS and the clients. '15' is optimized the number of clients per APS. However, if there is only one device with enough resource on the network, the APSM create another APS in this device.*

The *Session Manager* manages a client's sessions, sending session information to the others clients and requests allocation of an APS to the APSM. The *Context Manager* manages the context data in the Context DB, and the *Device Analyzer* analyzes the connecting client's device. The functions of the Device Analyzer are as follows: 1) distinguish between client's devices, 2) when the connecting client's device is mobile, the Device Analyzer requests allocation of an APS to the APSM through the Session Manager.

b) Client

The Client is a default user part, the architecture of which is organized by the Architecture Manager. The Client is loaded into Desktop Pc. When the client's resource is

sufficient, the architecture of Client is structured. In cases where the resource is insufficient, the Adaptive Proxy Server and Mobile Client are used. The main function of the client is videoconferencing. The Client consists of: 1) the Connection Manager, 2) Architecture Manager, 3) Monitoring Agent, 4) Adaptation Agent, 5) Context Agent, 6) Context Memory and 7) Communication Agent.

The Connection Manager is a component that manages the client's connection to the server and sends session information to the server. The Architecture Manager is a component that manages the client's architecture, which is composes of components according to the contexts. If its resources are insufficient for contents adaptation, the Architecture Manager organizes its architecture as that of a Mobile Client's architecture, at which point the Architecture Manager uses a runtime reconfiguration mechanism. Runtime reconfiguration can be performed by altering the connector bindings, as these mediate all component communication. The runtime reconfiguration mechanism has been proposed in many researches [9, 10].

The *Monitoring Agent* gathers the client context. The *Adaptation Agent* decides the method of communication and performs contents adaptation according to information in the Context Memory of other clients. The functions of the Adaptation Agent are as follows: 1) receive the context information of each client from the Monitoring Agent, and 2) determines which communication method is used by the client according to the context information.

The decision of the Adaptation Agent is based on the following policy:

[Monitoring Agent must have only one Parameter each step. (D)(W)(PO)(A) Parameter is Available, (D)(P)(PX)(A) Parameter is disabled.]

Decision Policy

```
If the first Parameter is (P), an application module
for PDA is required. The Default Parameter is (D)

If the second Parameter is not (W), but either (U), (L)
or (S), the Direct Audio Renderer and Capturer must
change. The Default Parameter is (W).

If the second Parameter is (C) or (M), an algorithm
change is needed. The Default Parameter is (M).

If the second Parameter is (S), the environmental con-
figuration of the client for this Parameter is (W),
change is needed.

If the third Parameter is (PX), a port change algorithm
is executed. The Default Parameter is (PO).

If the forth Parameter is (NA), an algorithm for syn-
chronizing the vector number with the connect number is
needed. The Default Parameter is (A).
```

The *Context Agent* stores the contexts to the Context Memory and obtains the contexts in the other client's Context Memory through the ACI (Agent Communication Interface). The *Context Memory* stores the client's context and communication methods. The *Communication Agent* communicates with the other clients, and consists of the *Transporter*, *Communication Method* and *Receiver*.

The *Transporter* obtains video and audio information from the system, and communicates this contents to the other clients using the Communication Method. *Communication Method* is a transmission code that is received from the communication target. The *Receiver* receives video and audio information from other clients.

c) Mobile Client

The Mobile Client is a client with an architecture for mobile users that have insufficient resources for a contents adaptation. The Mobile Client's architecture is decided by the Architecture Manager, which then organizes the Mobile Client's architecture. The Mobile Client is loaded into PDA or smart phone having video camera.

A Mobile Client has less functionality then a Client because the contents adaptation function is too heavy for a Mobile Client. Therefore, to support a Mobile Client's contents, adaptation is performed by an Adaptation Proxy Server.

d) Adaptation Proxy Server

The Adaptation Proxy Server is a proxy server, which performs contents adaptation in the place of a Mobile Client, which consists of 1) Context DB, 2) Context Agent, 3) Monitoring Agent, 4) Adaptation Agent and 5) Connection Manager. The APS is loaded Desktop Pc. If a Mobile Client connects to an APS through the Connection Manager, the Monitoring Agent gathers the context and session information of the connected Mobile Client, which are then stored to the Context DB. When a client is connecting to an APS, the Context Agent updates the Context DB in the Server. The Context Agent also obtains each client's Communication Methods using the session information of the Server. The Context Agent also sends the session information, context and Communication Method to the Adaptation Agent. The Adaptation Agent decides the appropriate Communication Method for contents adapted communication. The Adaptation Agent then composes the Communication Method for the Communication Agent of the Mobile Client and the Mobile Client then communicates with each client using this Communication Method.

In this paper, we have designed a reconfigurable architecture for a Mobile Client using the dynamic link in JAVA and Weaves Style. Weaves, made by The Aerospace Corporation, are networks of concurrently executing tool fragments that communicate by passing objects, and have a dynamic, object-flow-centric architecture designed for applications that are characterized by continuous or intermittent voluminous data floes and real-time deadlines. We have already addressed Weaves [8] in section 2. Fig.4 shows the concept of dynamic component composition.

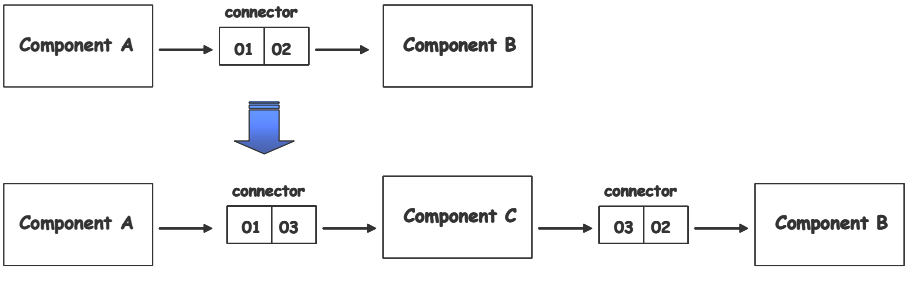


Fig. 4. The concept of Dynamic Component Composition

4 Implementation and Evaluation

The system proposed in this paper has been implemented based on JAVA SDK in Windows, Linux and Solaris environments. The main code for transmission of the picture information is made using JMF [4], for which the user needs a capturing device to obtain the picture information.

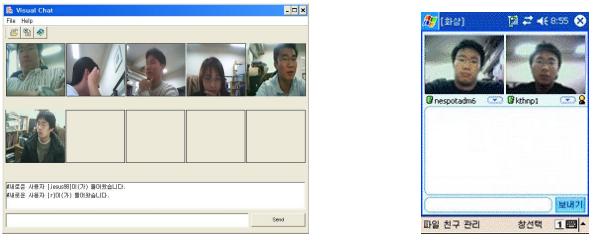


Fig. 5. Screen shot of our system

This system is implemented on a component basis, consisting of adjustable components, where each component is associated with an appropriate algorithm for adaptation to the user's operating environment as well as that of the overall network. This is the basic characteristic of self-adaptive software, which has been implemented using the Fractal [5] library for our proposed system.

Evaluation: According to the location of adaptation, an adaptation system is separated into the client-side, server-side and proxy server-side adaptation systems. [11] First, the client side adaptation system monitors and adapts the environment on its own side [12, 13], which could prevent the exposure of personnel information to the outside, and could also be suitable for performing reconfiguration of a control device configuration or to modify its functionality. Second, the server-side adaptive system monitors each client and performs contents adaptation, involving adaptive modules on their side. [14] Third, the proxy adaptation system has adaptive modules installed, which sets the proxy server between a client and server [15, 16], which can take advantage of being able to adapt, by the addition of the proxy server, without modification to the server or client components. In this paper, the difference between a general videoconferencing system and our proposed system, using the self-adaptive concept, has been evaluated, and the quantitative differences between the use of three adaptation mechanism - Server side, Client side and Proxy Server side adaptation - based approaches and a Dynamic adaptive architecture based approach. In Fig.11, the horizontal axis represents the adaptation time, and the vertical axis the number of clients.

In Fig.6, the use of the three adaptation mechanisms required more time than that with the Dynamic Adaptive Architecture. Evaluation environment is following.

Server : HP net Server. Os. Windows 2003 Server CPU 3.0G, RAM 1024Mb
Proxy Server : Os. Windows XP Pro, CPU 3.0G G RAM 1024Mb
Client : Os. Windows XP Pro, CPU. 2.4G RAM 512Mb

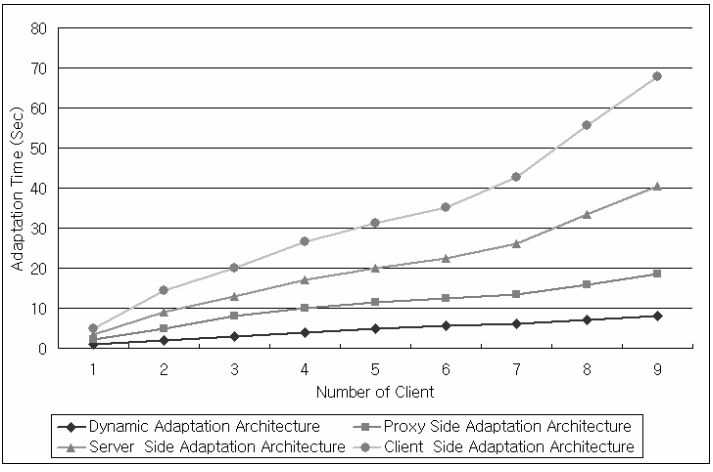


Fig. 6. Comparison of the adaptation and dynamic adaptation architectures when a mobile client is connecting. The use of dynamic adaptation architecture is more effective than those of the three adaptation mechanisms – Server side, Client side and Proxy Server side adaptation.

An almost mobile device has less resources for adaptation; therefore, a mobile device requires a proxy. However, if mobile devices have sufficient resources, they work more efficiently than if the client performs the adaptation themselves. In view of these facts, the use of Dynamic adaptive architecture is more effective. If the number of clients per APS is optimized, our system will spend less time on adaptation.

5 Conclusion and Future Works

In this paper, architecture based self-adaptive videoconferencing system, with dynamic adaptive architecture, has been proposed for internet based videoconferencing. Our system was able to adapt to the user's environment, requests and network environment. The users of our videoconferencing system are able to utilize a system that is optimized for their environment and network surroundings. Also, this system can apply as a self-configuring and self-healing system, because it is able to cope with new problems by the addition of new components. It also includes monitoring techniques that can deal with problems while the program is running.

In future work, this architecture will be applied to various multimedia systems and additional factors, which affect QoS, will be considered. In addition, the reconfiguration of software architecture will be focused on, and applied to adaptive multimedia systems.

References

1. R. Laddaga.: Active Software. Lecture Notes in Computer Science, Vol. 1936. Springer-Verlag, Oxford UK(2000)11–26.
2. Wenbiao Zhu, etc. al..: JQOS: a QoS-based Internet videoconferencing system using the Java media framework (JMF). IEEE ECE 2001. Vol. 1, 13-16. Canada (2001)625 - 630

3. Peyman Oreizy, etc. al..: An Architecture-Based Approach to Self-Adaptive Software. IEEE Educational Activities Department, Vol. 14 (1999) 54 - 62
4. Java Media Framework. http://java.sun.com/products/java-media/jmf/index.jsp
5. Pierre-Charles David etc. al.. : Towards a Framework for Self-Adaptive Component-Based Applications. Lecture Notes in Computer Science, Vol. 1936. Springer-Verlag (2003)1-14
6. Sung Doke Lee, etc. al..: Multi-agent based Adaptive QoS Control Mechanism in Flexible videoconferencing System. Advanced Communication Technology 2004. Vol. 2 (2004)745 - 750
7. R.N. Taylor et al..: A Component- and Message-Based Architectural Style for GUI Software. IEEE Trans. Software Eng., Vol. 22, No. 6, (1996)390–406
8. Michael M. Gorlick and Rami R. Razouk. The Aerospace Corporation..: Using Weaves for Software Construction and Analysis
9. Peyman. et al..: Architecture-Based Runtime Software Evolution. ICSE'98 (1998)
10. Quianxian Wang, etc al..: Runtime Software Architecture Based Software Online Evolution. IEEE CIMPSAC'03 (2003)
11. M. Margaritidis, G.C. Polyzos.: Adaptation Techniques for Ubiquitous Internet Multimedia. Wireless Communications and Mobile Computing, Vol.1, No.2, (2001)141-163
12. Brian Noble.: System Support for Mobile, Adaptive Applications. IEEE Personal Communications, Vol.7, No.1 (2000)44-49
13. C.Y. Hsu, A. Ortega, M. Khansari.: Rate control for robust video transmission over burst-error wireless channels. IEEE Journal on. Selected Areas in Communications, Vol.17, No.5, (1999)
14. Friday A., N. Davies, G. Blair and K. Cheverst.: Developing Adaptive Applications: The MOST Experience. Journal of Integrated Computer-Aided Engineering, Vol.6 Num.2, (1999)143-157
15. M. Margaritidis, G.C. Polyzos.: MobiWeb: Enabling Adaptive Continuous Media Applications over 3G Wireless Links. IEEE Personal Communications Magazine, Vol.7, no.6, (2000)36-41
16. IBM WebSphere® Transcoding Publisher, http://www-306.ibm.com/software/pervasive/transcoding_publisher
17. Gu, X.; Nahrstedt, K.; Messer, A.; Greenberg, I.; Milojicic, D.: Adaptive Offloading for Pervasive Computing. IEEE Pervasive Computing, Vol. 3.(2004)66 - 73

Contextual Interfacing: A Sensor and Actuator Framework

Kasper Hallenborg

Maersk Mc-Kinney Moller Institute,
University of Southern Denmark,
Odense M, 5230, Denmark
hallenborg@mip.sdu.dk

Abstract. The purpose of this article is to describe a middleware framework for interfacing sensors and actuators, which provides abstract and useful contextual information to applications. The framework is divided into three layers with different abstraction levels, and an integration layer. The framework is evaluated, exemplified, and further described by practical implementations using the framework.

Keywords: Sensors, context, abstraction, framework, pervasive computing

1 Introduction

Human perception is the mental processes, which deals with gathering and interpreting context information. In ubiquitous and pervasive computing, we face a similar task, when we try to establish a realistic relation between context and behavior of the application. Sensors are the analogy to human senses and the key sources for providing information from the environment. Interfacing of sensors and actuators is a common problem in application development. Each sensor and actuator has their own protocol and connection type, which makes it hard to implement a transparent interface from an application point of view.

For human-interactive, context-aware applications, the response time for a sensor or actuator has lower priority, thus the number of abstraction layers could be increased for the interface. With an increased number of layers in the interface, it is easier to reuse and generalize specific parts of the interface scheme.

2 Related Work

Dealing with sensors is not a new trend in software applications, but the number of sensors providing information to the applications has increased rapidly over the last years, and they are a must for creating the killer applications of pervasive computing, which truly will revolutionize our impressions of computers, if they follow the spirit of Mark Weiser's *Calm Technology* [19]. Thus, abstracting and interpreting sensor information is and will be a crucial task of pervasive

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 846–857, 2005.
© IFIP International Federation for Information Processing 2005

applications, which means that standardizing sensor integration and context interpretation will be an important part of most applications.

Anind Dey's *Context Toolkit* [1] is among the most cited works in this area. The Context Toolkit hides the details of the sensors behind *context widgets*, which encapsulates the code for communicating with the hardware sensors, and provides a standardized API for retrieving the data.

Another approach is taken by Schmidt et al [16,17], who in a layered architecture hides the context retrieval process. It consists of three elements; *Sensor*s, which output is regarded as a function of time, *Cue*s, which combines and abstract the current and previous data from a particular sensor, the *Context* layer combines the input from all Cues to a current situation of the context, and finally the *Application* layer holds mechanisms to perform actions, when entering, leaving, and operating in a given context state.

Other approaches with focus on contextual integration and abstration are the CALAIS architecture [10], the Ektara [14] by MIT Media Lab, the Context Information Service (CIS) proposed by Pascoe [12]. More recently, Hackmann et al [13] have proposed a simple pattern-like architectural middleware (CONSUL) for providing context-awareness to application.

3 Approach and Overview

Basically, the framework is best described as a middleware software component with the purpose of adapting between hardware sensors and actuators, and context-aware applications. As a consequence of the finite computing capabilities, interfaces to the analog world can to some extent always be regarded as discrete. Typically we sample the environmental variable at a random point in time and regard the output as valid until the next sample. As the application domain of the framework is intended for context gathering in human-like perception processes, the intelligence built into the systems elsewhere should make the systems fault-tolerant to a few missing values from the environment. Thus *synchronization* must be the first aspect of dealing with sensors and actuators. In the layered model of the framework, figure 1, the synchronization layer is illustrated at the bottom, controlling the directed access with hardware interfaces of the sensors and actuators.

Requesting a few sensors values is far from the fascinating illusion of tomorrow's pervasive systems, and responding to changes of a single environmental variable is also on the edge of the definition of being context-aware. Similar to human perception, we would require the ability to combine various stimuli from different sources to a whole representation of a single impression. The next level of the framework is named *aggregation*, and covers not only combination of data from various sensors, but also aggregating multiple data from the same sensor to a single value, such as averaging over the last 2-3 values from a temperature sensor. Combined sensors do not necessarily have to provide mutual exclusive information, for instance a device could be equipped with two or more positioning

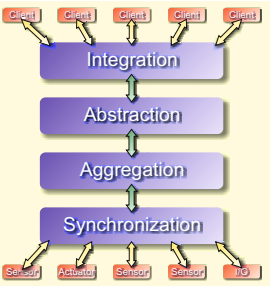


Fig. 1. Layered model of the sensor framework, composed by the four required components; Synchronization, Aggregation, Abstraction, and Integration

sensors, which could be combined and prioritised to a single position, based on the situation of the device, such as indoor and outdoor.

Raw sensor data give little or no meaning to most users, and some information from the sensor might be irrelevant, such handshaking and status signals. E.g. GPS signals contains a lot of information, which is irrelevant, if only a "best guess" of the current position is requested. Beside from a second option to filter data, the *abstraction* level of the framework allow the user to convert the data to more meaningful information.

Last but not least, integrating context information into applications is perhaps the biggest challenge towards a generic approach for using context. Several previous attempt for unifying context integration exists, such as a generic context server [15]. We will provide a number of standardized interfaces for integration with existing technologies. Summarizing,

Synchronization grasp the current data from deployed sensors, or forwards command to actuators.
- General interfaces for standard purpose I/O interfaces.
- Support interest from multiple clients.

Aggregation provides means for combining primitive and related sensor data to a joint structure expressing a more cohesive stimuli.
- Unlimited and unconstrained means for combing all kinds of sensors.
- Parts of the structure could be reused for different purposes.
- Option to prioritize between overlapping sensors or sensors with common characteristics.

Abstraction interpreting the raw sensor data to more meaningful information for both humans and applications.
- Provide a common data structure for easing the integration of sensor and actuator information with applications.
- Provide a set of common structures for general context information, eg. location data.

Integration provides means for integrating the sensor framework with existing software technologies.
- Support a number of general technologies for integration with applications and services.

4 Synchronization

As stated above the purpose of the synchronization layer is to retrieve or send data from or to the hardware or logical devices, respectively. Retrieving data from sensors could either be initiated by frequently polling by the client or the interrupts from the hardware sensors sending event notifications through the interface. As one of the design criteria is to support a broad range of client functionality both mechanisms must be supported. Thus, besides the general methods for requesting data from a higher level in the framework hierarchy, the synchronization level should also be able to generate events and forward them upwards. Monitoring the outdoor temperature is a typical example of polling the sensors now and then, whereas triggering contacts in an alarm system justifies the need for event generations as well. The framework must support both methods, even it is commonly agreed that the notification model is preferable with respect to performance, but for some applications the setup of listeners, event generators, and event handlers requires an inappropriate amount of resources.

The other important functionality of the synchronizer is to support multiple clients' interest in the same data. The usual approach would be to discard the data after they have been retrieved by a client, and let the client to decide, which data needs to be stored. Supporting multiple clients raises the dilemma of making the same data available to all clients or regards each invocation from clients as a single request. As the first approach is the most general, we will go for that solution, as the second approach easily could be implemented on top of that by discarding all current data and request a new value.

4.1 Multiple Client Support

The second dimension of customizing the synchronizers is the support for multiple clients. It is not mandatory for a synchronizer to support multiple clients, based on the arguments of constrained devices and simple applications with no required support. Transparency should also be maintained, so a simple synchronizer could be substituted with a multi-client synchronizer, without affecting the existing code. It has one strong implication for the interface of the synchronizer, as most object-oriented programming languages do not provide access to the caller from the callee, without a passed reference to the caller. Identification of the caller is required in order to select and return the right data, and to clean up and maintain the repository of stored data.

4.2 IO and Logical Devices

Communicating with the real devices is one of the boundaries, where the framework has to fit with existing technologies and defined standards we typically cannot change or influence, both caused by physical limitations and the extensive distribution of sensors. The interface to the devices should be kept simple for the sake of generality; basically we are only interested in sending and receiving a chunk of data to and from the device. Configuration of ports and other details of

the communication protocol, settings, etc. should be hidden in the implementations of the interface, and would not be under control from the synchronizers, as transparency should secure that synchronizers would not care about the device is connected via a serial port, USB, WLAN, or a file-object for demo purposes.

5 Aggregation

Grouping information is, if not explicitly stated, a commonly accepted way to cope and deal with excessive amounts of information, if not the only one. More or less consciously we do it all the time, and it helps us to hide the unimportant details for a particular matter. Instead of describing everything by their atomic components, a cumulative identifier simplifies the model. In a normal conversation almost anybody would know, or a least have an idea of, what you mean by a house. So instead referring to four walls of bricks and windows, and a roof, where bricks are small boxes of burnt clay, windows are rectangular boxes of glass with a frame, which is ... , etc. Instead we have this composite identifier of a house, which makes conversation possible, but we have to accept that the other person do not necessarily picturize the same house, if not described further.

From this example it is also clear that various components, which are part of the complete composite structure, may themselves be identified entities of importance to others or in another situation, such as the window being a window without carrying about being part of a house. In this way we have different component or details of the composite, which has certain interest for different people or purposes, which again could include parts that are important to others.

So the purpose of the aggregation layer is to collect and present data from synchronizers in a meaningful way for the abstraction layer. The intension is to relate all data, which are required to construct a client-ready data structure in the abstraction layer, by only referencing a single aggregator.

Data from the aggregator may still be represented in the form they are extracted from the synchronizers, but a conversion could optionally be applied, such as error correction and filtering, corresponding to the nature of human perception, where only a few out of millions stimuli reach the memory matching processes that defines our behaviour.

Finding a data structure to hold the retrieved information in not an easy task, as both type or name of the data, and the data itself should be accessible. The approach taken is to compose a `Map` structure, where combined information from one or more sensors could be separated to individual entities in the mapping.

An aggregator is intended to communicate with only one synchronizer, making event propagation much easier, so aggregators must be able to combine their data to a single entity, which can be addressed from the abstraction layer. The *Composite* pattern [4] is a simple mean to combine aggregators in an unlimited tree-structure and keep a simple interface to the abstractors, which cannot tell if a complex structure or just a single aggregator is responding.

A `CompositeAggregator` provides a simple approach for combining data from a list of children into a single mapping. Composite aggregators could com-

bine different positioning measurement into single location information. Consider the case, where your device has two means to determine the location, such as a GSM phone, which could the determine the position based on the cell-ID or even a more advanced triangulation approaches. The phone could in addition be equipped with some sort of Bluetooth means for accurate locale indoor positioning.

The two methods of the `Aggregator` interface return and set the translator of the `Aggregator`, respectively. By this mean, it is extremely simple for the user to customize the `Aggregator` for a specific purpose, only an `AggregatorTranslator` has to be provided, converting data to and from the aggregation layer.

6 Abstraction

10.3 kΩ of a 20 kΩ LDR or NTC resistor may give sense to some, but most people would have no clue, and even technical skilled would have no idea of how it relates to lightning and temperature conditions, respectively. It depends very much on the scale and linearity of the sensors, just to convert the data into more familiar scales, such as *Lux* and *Celcius*. Or consider the following output from a sensor

```
$GPRMC,095211.808,A,5524.2563,N,01024.0460,E,2.03,324.78,230702,,*03
$GPGGA,095212.808,5524.2568,N,01024.0448,E,1,04,4.4,45.8,M,,,,0000*3F
$GPGSA,A,3,20,11,01,07,,,,,,,,,7.9,4.4,6.6*38
```

you cannot gain much information from the listing above, if you are not familiar with the NMEA 0183 standard used by the GPS sensor.

There is no need for software designers to deal with these kind of low-level details, every time they request interaction with the environment, and therefore have to interface sensors. By abstracting the sensor specific information, we can bring sensors and actuators into play, with only little knowledge of the underlying hardware.

The purpose of the layer is to compose appropriate and client-ready data based on the data retrieved from the aggregation layer, and send it to clients through the integration layer. No further conversion or adaptation of the data should be performed in the integration layer, just integration and interfacing with various devices or protocols should be handled. Data can either be retrieved or put through the abstraction layer by the methods `getData` and `putData`, respectively.

Translators are used to convert data to and from the abstraction layer, in the same manner as the aggregation layer. The `AbstractorTranslator` is, similar to the `AggregatorTranslator`, defined as a static inner interface of the `Abstractor`. As for the aggregation layer, the purpose of using translators is to simplify the framework for the user, so just a simple translator implementing the conversion methods have to be provided in combination with the default `Abstractor` implementation.

Sticking to the focus of the framework as middleware for ubiquitous systems, one sensor (hardware or logical) would rarely be enough as the systems evolve.

The intension is to let the framework compose the full context conditions the application should respond to. Instead of the application composing its own weather condition from e.g. a temperature and rain sensor. Separating the aggregation and abstraction allow us to have different abstractions of the same aggregated structure. Abstractors could reference any node in the tree formed by the aggregators, so parts of the tree could have it own interpretation for one or more clients.

7 Integration

You can gather and interpret all the information you want, but if you do not make them available to users and try to fit their requirements for collecting such information, you will never reach more than your own test applications. Thus at the integration level of the framework, we have reached the upper boundary of the middleware, where we have to fit to existing standards in order to associate with most users, and this is not an attempt to be just a good sales guy. We could easily wrap the data and drop them into our own context server, but first and foremost that have not been the focus of the work conducted on the middleware framework. On the other hand there is a number of existing means to integrate or provide context information to users, and as closer we approach a general standard des easier will the integration be.

The intension of the integration layer is not further to manipulate or abstract the data received from the sensors, just to make is easier to provide the data in various ways, such as services in networks or integrated directly into applications. Our initial and main focus for integration have been the Jini^{TM} Technology, therefore the standard implementation provides default implementation of integrators and adaptors for the Jini community, but working and experimenting with other networking architectures or similar service systems, will constantly extend the number of provided adaptors and supported integration methods. By this mean integration with the Elvin Messaging system [2] has been implemented as further described in the sections of the examples.

Integrating directly with applications would typically be in terms of binary Java objects or standardized data structures, such as XML documents. Following the intensions of the framework these data should be provided in abstract and general formats from the abstraction layer, and the integrator might only have to wrap these information and make them accessible for the applications, for instance in terms of Java Bean objects.

8 Examples

We will look at three examples for testing different aspects of the framework, which might look rather independent, but in the end they will be grouped together to finally test those aspects of the framework.

The first example is to integrate a GPS sensor and provide positioning information in a general format and make it accessible to interested clients. The

example is a simple test of the most basic functionality of the framework, which will illustrate the minimal work that have to be done by the user. The next example is controlling a RCX brick from a PDA. Abstract and human-friendly commands should be translated and send to the RCX brick, for control of attached sensors and motors. The last example is the integration of the Smart-Its, a general sensor-board with multiple sensors, that will focus on testing the aggregation layer of the framework. Finally, all three examples will be combined to a single robot with a control part consisting of a PDA and a RCX brick, which will serve the purpose of testing reuse and combination of several independent setups in the framework.

9 Example: Interfacing GPS

Location is arguably the most important contextual information for most pervasive and ubiquitous applications, thus interfacing a positioning device is a relevant example for testing the framework.

A number of different positioning devices exist, but besides cellular-information of mobile phones, the GPS (Global Positioning System) is by far the most prevalent. Most common receivers use a simple serial protocol based on the NMEA sentences [11], so these sentences have to be parsed and combined in order to give some useful information for the application layer.

The basic purpose of the test is to retrieve and interpret data from a GPS receiver, which basically could be done with any GPS receiver that allow us to access the NMEA sentences. The challenges of deploying it to a PDA and let it control the receiver and framework components are minimal and not given any special focus during this test, so it could be mirrored to a laptop without any changes, except for a recompilation of the native part.

One way to provide location information in a general structure could be the SLO Protocol [7], which is one of the implemented standards in the framework to demonstrate the implementation of abstraction translators. It extends a standard XML translator that sets up all the basic for producing XML documents.

10 Example: Interfacing Lego Robots

In the recent years Lego has merged computer power and the standard bricks to enhance the intelligence of the toys. Special bricks can be programmed with some functionality. The focus of this example is to send simple immediate commands to the firmware in the RCX brick, such as start and stop of a motor, setting the speed of the motor, retrieving sensor inputs, checking the battery level, or playing a sound.

The simple approach is to control the RCX computer via the IR-tower, which is part of the Lego Mindstorm set. Our main objective is to control the RCX through an IR port of a PDA, therefore additional `IODevice` implementations have to be provided for the PDAs. We agreed on a setup, where a PDA with a built-in IR-port is mounted on the robot and the IR ports of the PDA and the

RCX are aligned. The PDA would be wireless connected to the main computer using wireless LAN. The protocol for communicating with the brick is somewhat challenging to the framework, because RCX opcodes and messages are wrapped into packets with headers and checksums. To complicate it even more, after each byte send (beside the header) the same byte must be resend negated.

Using the IR port of a PDA is a lot more complicated than using a serial connection, as we have to take control of the hardware pins on the controller boards using special registers of the PDA. We have successfully implemented communication with Cassiopeia E-200 and IPAQ H5550 PDAs.

10.1 Aggregation and Abstraction

In the aggregation layer we handle the translation of RCX requests and commands into real RCX messages with header and checksum, and responses from the RCX are unwrapped, and the information is provided in a mapping structure to the abstraction layer. It is also the task of the aggregator to check messages and responses from the RCX, that they fulfil the required and expected format.

For the example we have implemented two abstractors, one which represent a Jini service object holding the full state of the RCX. The other abstractor generates Tickertape [3] messages for the Elvin messaging system, which are presented in a Tickertape application, where only the responses are translated into messages and commands.

11 Example: Interfacing Smart-Its

The Smart-Its from Lancaster University consist of a core board, which contain power sources, a programmable microcontroller, memory, and a wireless communication chip. The core board has a socket for add-on boards, which in our case is used for the general sensor add-on board containing 5 different sensors; Light sensor, motion detector, touch sensor, temperature sensor, and dual axis accelerometer.

Tailoring the Layers. Retrieving the information from the Smart-It basestation is extremely easy, as the standard code sets up the Smart-Its to put the sensor data over the serial port line by line. Thus, the code up to the aggregation level would be identical to that of the GPS receiver, except for the com port settings of the `IODevice`; 115,200 bps, 1 stopbit, 8 databits, and no parity.

Beside the important task of retrieving the data for the sensors through the synchronizers, the aggregator has to compute these changes based on the history of sensor data.

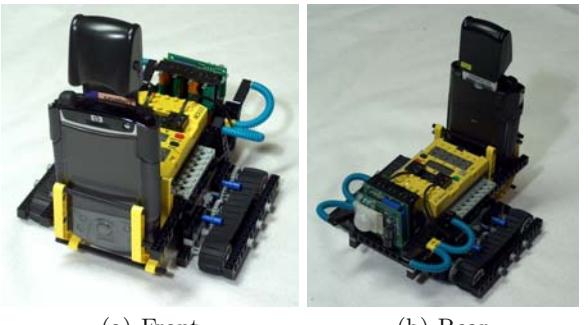
Aggregation Layer. The Smart-It sensors provide two types of values, either booleans from touch and motion sensors or integer values from the other sensors. The integer values can be considered as a discrete function of e.g. time or number of measurements, thus we can apply numerical methods to compute

properties for a similar or fitting continuous function, which is much easier to handle computationally and relate to, and predictions of future values could also be possible. In this example we will only focus on a number of simple properties for the function that can be useful to determine how extensive a change in the measured value is.

Abstraction Layer. Abstracting data from a general sensor board as the Smart-It to a united context descriptor is almost impossible for a general purpose, as the domain of context-aware application is so broad. We could tailor the abstraction of the Smart-It's data, to collect environmental context information for the robot only. Similar to the data abstracted from the GPS sensor to an XML document of the SLO protocol, we have defined an XML scheme with generalized Smart-Its data, holding the current values of the sensors and optionally the changes to the state of each sensor.

12 Example: Putting It All Together

It may look like a weird artefact with superfluous technologies attached, but the challenge was to compose an artefact with various capabilities from existing technologies.



(a) Front (b) Rear

Fig. 2. The SmartRobot with the RCX computer, a IPAQ H5550, GPS receiver, and a Smart-It attached

To give a brief overview of the application of the combined technologies in this example, a Lego robot will drive around and report various information about the environment, it is passing through, and it can be controlled by the user via simple commands. We will use the TickerTape messaging board of the Elvin message system to present these reports to the user, and allow the user to control the RCX robot via simple commands. The Smart-It sensor-node is also attached to the robot and a base-station is attached to a host computer, which runs the example of the Smart-It and produces a XML document that is made

available in a repository accessible by the IPAQ via WLAN. By this means the XML document becomes a source of input to the robot, which could be *sensed* by an XML parser.

12.1 Compose the Primitive Aggregators

We have chosen the easy approach of combining primitive aggregators that each represent a channel of context information from lower layers. The standard `CompositeAggregator` simply merge the mappings of child aggregators to a single mapping forwarded to the abstractors, ignoring overlaps in keys (last child overwrites first). Extending this algorithm prioritizing between identical or similar context information could be introduced, e.g. for the robot we could exclude the light sensors of the Smart-It, if light sensors were attached directly to the RCX or reverse.

12.2 Chatting with Elvin^{TM}

The objective of the SmartRobot is not to do any fancy tool work in the field, but just wandering about and report experiences from and about the environment it is passing through. The application can be seen as an extension of the RCX example, where commands are sent back and fourth between the RCX robot and the TickerTape message-board of the Elvin messaging system. Instead of simple and informative command-like messages, we have extended the abstractor to compose a number of more descriptive messages, that can report about changes in the environment, e.g. the temperature decreases/increases, and GPS information can, when available, be used to set a picture in memory of the current state, so messages could sound like:

> *... last time I was near this location, it was warmer and lighter, but my accelerometers tells me that I'm still not shaking ...*

13 Conclusion

We have designed and implemented a general framework for interacting with sensors and actuators. The default implementation and the core framework, provides a number of way to setup the communication with the hardware, and retrieval of data in particular. We support asynchronous multiple-client interests of the retrieved information. Customization of the hot spots of the framework in easily customizable through simple translators, which translate up and down between the data structures of the different levels in the framework. We have implemented a couple of examples to test and validate various aspects of the framework with great success and only minor changes to the core framework. We hope the supported technologies constantly will grow through different examples with the framework. We will continue working on the framework structure and enhance some of the provided tools for better and easier integration with various applications.

References

1. A. K. Dey. *Providing Architectual Support for Building Context-Aware Applications*. PhD thesis, Georgia Institute of Technology, November 2000.
2. DSTC. Elvin - content based messaging. http://elvin.dstc.edu.au/.
3. DSTC. Tickertape. http://elvin.dstc.edu.au/projects/tickertape/.
4. E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns - Elements of Reusable Object-Oriented Software*. Addison Wesley, 1995.
5. H. Gellersen. The smart-its project. http://www.smart-its.org/, 2001.
6. J. Hightower, B. Brumitt, and G. Borriello. The location stack: A layered model for location in ubiquitous computing. In *In Proceedings of the 4th IEEE Workshop on mobile Computing Systems & Applications (WMCSA 2002)*, pages 22–28, Callicoon, NY, USA, June 2002. IEEE Computer Society Press.
7. M. Korkea-aho. *Location Information in the Internet*. PhD thesis, Helsinki University of Technology, October 2001.
8. Lego^{TM}. The offical lego mindstorms website. http://mindstorms.lego.com/.
9. L. Merk, M. S. Nicklous, and T. Stober. *Pervasive Computing Handbook*. Springer Verlag, January 2001.
10. G. J. Nelson. *Context-Aware and Location Systems*. PhD thesis, Clare College, University of Cambridge, United Kingdom, January 1998.
11. The national marine electronics association (nmea). http://www.nmea.org. NMEA 0183 standard.
12. J. Pascoe. *Context-Aware Software*. PhD thesis, Computing Laboratory, University of Kent at Canterbury, August 2001.
13. G. H. C. J. J. Payton and G.-C. Roman. Supporting generalized context interactions. In *Proceedings of the 4th International Workshop on Software Engineering and Middleware*, Linz, Austria, September 2004.
14. A. S. P. Richard W. DeVaul. The ektara architecture: The right framework for context-aware wearable and ubiquitous computing applications. Technical report, The Media Laboratory, MIT, February 2000.
15. D. Salber and G. D. Abowd. The design and use of a generic context server. Technical Report GIT-GVU-98-32, Georgia Institute of Technology, GVU Center, College of Computing, 801 Atlantic Drive, Atlanta, 1998.
16. A. Schmidt, K. A. Aidoo, A. Takaluoma, U. Tuomela, K. V. Laerhoven, and W. V. de Velde. Advanced interaction in context. In *Proceedings of the first International Symposium on Handheld and Ubiquitous Computing (HUC99)*, volume 1707, pages 89–101, Karlsruhe, Germany, 1999. Springer.
17. A. Schmidt and K. V. Laerhoven. How to build smart appliances. *Personal Communications, Special Issue on Pervasive Computing*, 8(4):66–71, August 2001.
18. A. M. R. Ward. *Sensor-driven Computing*. PhD thesis, University of Cambridge, August 1998.
19. M. Weiser and J. S. Brown. The coming age of calm technology. Technical report, Xerox PARC, October 1996.

MDR-Based Framework for Sharing Metadata in Ubiquitous Computing Environment

O-Hoon Choi, Jung-Eun Lim, and Doo-Kwon Baik

Department of Computer Science and Engineering, Korea University,
SungBok Gu, Anam Dong, 135-701, Korea
{pens, jelim, baik}@software.korea.ac.kr

Abstract. To resolve the syntax, structure and semantic heterogeneity for sharing information resources, the representative technologies are XML and Metadata. XML is used to represent the syntax and structure of information resources but the various XML schema definitions that have been developed by independent organizations without any standards or guidelines, make it difficult to share the semantic meaning of XML encoded information resources. In this paper, we propose a mechanism, named MSDL that represents the exact meaning of XML tags by describing the structural and semantic differences with standard metadata in metadata registries. MSDL overcomes the limitations of other approaches with respect to exactness, flexibility and standardization, and provides an environment for business partners using different metadata to share their XML encoded information resources.

1 Introduction

XML is a standardized text format designed specifically for transmitting structured data to web applications. With XML, application designers can create sets of data element tags and structures that define and describe the information contained in a document, database, object, catalog or general application, all in the name of facilitating data interchange [1]. XML documents, the unit of data exchange, are based on schemas (XML schema definitions). XML schema is a specification defining the structure of tags and plays a role of metadata on XML documents. Metadata is a data about data. And it is essential to information sharing. It describes attribute, property, value type and context on data.

One important point is that XML documents based on same schema definition have same structure and tags. Therefore, it is possible to share data between applications if they use standardized schema definition. Such standardized schema definitions include CML DTD [2] used in chemistry domain, Math DTD [3] in mathematics, ONIX [4] in digital library domain, and so on. So, common schema definition makes it possible to share tag elements and attributes defined in schema definition, and to understand semantics of the XML documents.

However, in many cases, each system (organization) has developed their own schema definition and use XML documents based on the schema. There is no interoperability with other systems without manual converting processes.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 858 – 866, 2005.
© IFIP International Federation for Information Processing 2005

For example, if we represent "author" of a book named "Database Design", Dublin Core [5] which is a metadata standard for describing general properties of information resources, uses a term "creator", but ONIX [4] which is a metadata standard for digital contents, uses a term "author"

In this paper, we first categorize heterogeneity between XML documents – syntactic, structural and semantic heterogeneity, and second design a framework (FSMI, Framework for Sharable Metadata Interoperability) to increase the interoperability of XML encoded information resources using different metadata sets. In this framework we propose a mechanism, named MSDL (Metadata Semantic Description Language) that represents the exact meaning of XML tags by describing the structural and semantic differences with standard metadata in metadata registries. With MSDL, users can describe the syntactic, structural and semantic difference of their XML schema to other's schema, and based on this description, can convert other system's XML document to a document with their own schema.

2 Related Works

In this section, we introduce some other approaches that enable the interoperability of XML documents having different schema definitions. The most straightforward approach for XML document interoperability is 1:1 mapping. In 1:1 mapping, each system tries to map their XML schema to other's schema. Microsoft's BizTalk [6] is the representative software. Users of BizTalk create a mapping table for each other application, and interchange their XML documents. After receiving, it converts the documents according to receiver's understandable format.

Maintaining integrated global schema is another method. EAI solutions analyze each application's data and processes, create global mapping tables and rules, and enable each system to exchange XML encoded data. Also, many organizations develop standard metadata (XML schema) and member of the domain use it as a data interchange format. The representative examples are ebXML [7] and Rosettanet [8] in electric commerce domain, and ONIX in digital contents (e-book) domain. This approach is very simple and powerful when all members use it as their data standard.

These three approaches have advantages and disadvantages in exactness, flexibility, initial cost and maintenance cost. In the last section, we will compare our proposed approach to them.

3 Metadata Model

We use data element model of ISO 11179 as a standard metadata model for comparison. Data element is a basic unit of identification, description and value representation unit of a data.

Definition 1 (Data Element). *Data Element is a basic unit for definition, identification and representation of data. It consists of data element concept and permissible value domain.*

$$DE = \{(dec, r) | dec \in DEC, r \in R\}$$
$$DEC = \{(oc, p) | oc \in OC, p \in P\}$$

The combination of an object class and a property is a data element concept (DEC). A DEC is a concept that can be represented in the form of a data element, described independently of any particular representation. In the examples above, annual household income actually names a DEC, which has two possible representations associated with it. Therefore, a data element can also be seen to be composed of two parts: a data element concept and a representation. Data element is composed of three parts as Figure 1 [9].

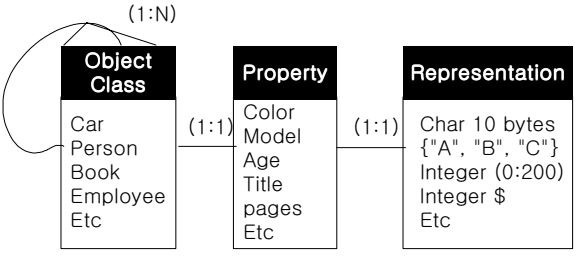


Fig. 1. Structure of Data Element

Object class (OC) is a set of ideas, abstractions, or things in the real world that can be identified with explicit boundaries and meaning and whose properties and behavior follow the same rules. Examples of object classes are cars, persons, books, employees, orders, etc. Property (P) is what humans use to distinguish or describe objects. Examples of properties are color, model, sex, age, income, address, price, etc. The most important aspect of the representation (R) part of a data element is the value domain. A value domain is a set of permissible (or valid) values for a data element. For example, the data element representing annual household income may have the set of non-negative integers (with unit of dollars) as a set of valid values. Figure 2 shows some examples related on Definition 1.

OC={person, car, employee, book, product, ...}
P={name, age, title, cost, ...}
R={char 30byte,{blue, red, green},{0..100},...}
DEC={personname, carname, booktitle, productcost, ...}
DE={(personname, char 30byte), (carname, char 30byte), (productcost,dollar),...}

Fig. 2. Examples on Data Element

A metadata registry (MDR) supports registration, authentication and service of such data elements. Each data elements have a unique identifier, all users and programs using the metadata registry identify a data element by the identifier. A data element also contains it definition, name, context, representation format, and so on.

4 MSDL

4.1 Information Heterogeneity

Table 1 shows the type of heterogeneity between standard metadata based on ISO 11179 MDR and XML schema definitions. Semantic heterogeneity means that the two xml tags has same meaning but different names. Structural heterogeneity means that two sets of xml tag have same meaning but different structures including composition, decomposition and rearrangement. Syntactic heterogeneity means the difference of data values in XML tags. Conflicts in this heterogeneity include measurement unit, data type and data code sets.

Table 1. Heterogeneity of XML Documents

	Description and Examples
Semantic Heterogeneity	Different terminologies have same semantics in context. ex) title – book name, price - cost
Structural Heterogeneity	Structurally different with same semantics ex1) name - first name, last name ex2) first name, last name – name ex3) last name - first name, last name
Syntactic Heterogeneity	Semantically, structurally same, but have different data type, measurement unit and code sets ex1) {kr, jp, cn} - {korea, japan, china} ex2) mile - kilometer ex3) integer - float

4.2 MSDL

The key idea of this paper is that users of each system describe the difference between their XML schema and standard metadata registered in metadata registry. Then they interchange their XML documents with other system as resolving the difference of other's document using the descriptions.

We propose MSDL (Metadata Semantic Description Language) as a tool for describing the difference between local XML schema and standard metadata in MDR.

MSDL consists of namespace on the location of MDR and local schema, correspondent information (MAP) and transformation rules for converting local document to/from standard document.

Definition 2 (MSDL). *A language for describing the difference between local schema and standard metadata. It uses XML syntax.*

MSDL= (mdrNameSpace, xmlNameSpace, MAP, CODESET)

Where, mdrNamespace and localNamespace must appear once in a MSDL document, but MAP appears as much as the number of tags to be converted.

Definition 2.1 (MAP). *MAP defines semantic, structural and syntactic conflicts between tags in XML documents and metadata in MDR.*

$$MAP=\{(mdrE, localE, maptype) | mdrE \subset DE, localE \subset XS, maptype \in MapT\}$$

Where, mdrE is a subset of standard metadata in MDR, and has one or more elements. localE is a subset of metadata defined in local XML schema, and also has one or more elements.

Definition 2.2 (Mapping Type). *MapT defines structural conflicts and syntactic conflicts between metadata with same semantics.*

$MapT=\{(sct, vct \mid sct \in SCT, vct \in VCT)$
$SCT=\{substitution, composition, decomposition, rearrangement\}$
$VCT=\{(CodeSetmdr, CodeSetlocal), (MUnit, cvalue), (TypeCasting, cfunction)\}$

Where, number(CodeSetmdr) = number(CodeSetlocal),
Munit = measurement unit (m, km, kg, etc.),
Cvalue = constant for measurement unit change,
cfunction = name of system support function for type casting

SCT (Structural Change Type) has one of the four elements – substitution, composition, decomposition, and rearrangement. Where, substitution is for resolving semantic conflict and other three elements are for structural conflict. For example, "book title" and "book name" have same meaning but different name, in this case, "substitution" is applied. On the other hand, a "name" tag may correspond to two tags "first name" and "last name" In this case, "composition" or "decomposition" is applied.

```
<map>
 <mdrElement>
  <DataelementName>
     <element>Price</element>
  </DataelementName>
  <elementID>DE024041</elementID>
  <dataType>
    <nonCodeSetDataType>string</nonCodeSetDataType>
  </dataType>
  <measurementUnitID>MU000000</measurementUnitID>
 </mdrElement>
 <localElement>
  <elementName>Price</elementName>
  <elementPath>/BooK/Price</elementPath>
  <dataType>
   <nonCodeSetDataType>string</nonCodeSetDataType>
  </dataType>
  <measurementUnitID>MU000001</measurementUnitID>
 </localElement>
 <mappingRule>
    <mappingType>substitution</mappingType>
 </mappingRule>
</map>
```

Fig. 3. An Example of MSDL Document

VCT (Value Change Type) has one of the three elements – code set conflict, measurement unit conflict and data type conflict. In code set conflict, we consider only the case of two code sets having same number of elements. A MSDL document must have one or more mapping type to describe the difference between standard metadata and local schema. Figure 3 shows an example of MDSL document mapping local element "Book price" to standard metadata element "Price" In this example, two elements have semantic conflict and syntactic (measurement unit) conflict.

4.3 XML Conversion Process Using MSDL

MSDL just specifies information about conflicts of local schema and standard metadata. Thus, it is necessary to develop a system which transform imported XML document to self-understandable one using MSDL. We call this system as FSMI (Framework for Sharable Metadata Interoperability). FSMI is a system for transforming XML encoded source data to fit other system's XML document using each system's predefined MDSL documents. Figure 4 shows the transforming process.

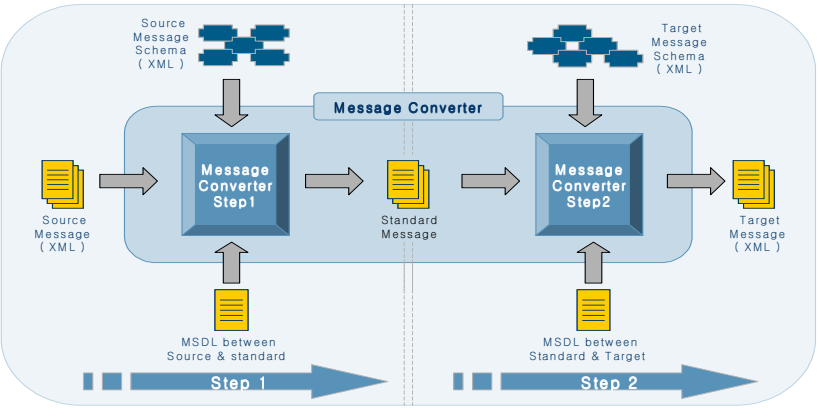


Fig. 4. XML document transform process

XML document converting process consists of two steps. The first step is to generate a standard document with standard data element in MDR from source document, local XML schema and MSDL of the schema. Then, in the second step, standard document is converted to target document using target document's XML schema and MSDL.

In the step 1 and step 2, FSMI applies mapping rules in MSDL to resolve structural and syntactical heterogeneities (conflicts). Generally, these two types of heterogeneity co-exist in a document, it is necessary to apply mapping rules with combined manner.

We will apply two steps mapping rule to solve discordance in document translation processing. In the first step, we translate a source document to a standard document that is structured by standard data elements using source document, its XML schema, and it's MSDL. In the second step, we translate a standard document to a target

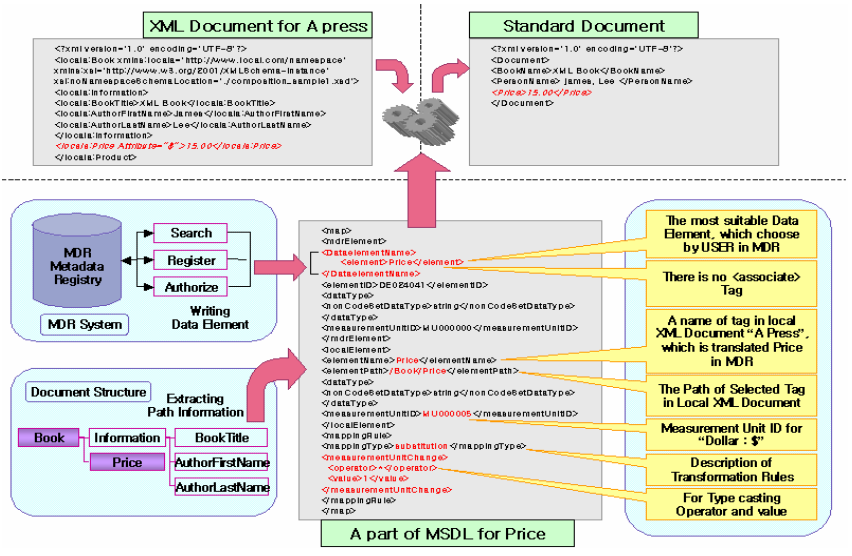


Fig. 5. Translation using MSDL

document that is used by other system using target document's XML schema and it's MSDL. In each step, we apply translation rules that are described in MSDL. It is need to apply lots of mapping rules to solve discordance of representation and structure. With two steps document translation process, we get a compound mapping rules. Figure 5 shows an example of translating a source document to standard document using MSDL.

5 Comparison

1:1 mapping approach is relatively exact because experts of each system directly map to other system's schema. But, the mapping cost increases exponentially as the number of related systems. Especially, in case of schema change, the maintenance cost will be high. Global schema can cover large parts of each system's characteristics of schema through analysis process, and the maintaining cost is very low in case of schema change because systems only have to change related parts of global schema. But the initial building cost of global schema may be high.

If all systems use standardized XML schema, then there are no syntactic, structural and semantic conflicts. It is very ideal status on data sharing and integration. But It is impossible to make a standard system (schema) that support all the case of system, and necessary to consider another processes for data that standard does not cover.

Also, a domain may have more than one metadata standards. For example, in digital content domain have metadata standard including ONIX, MARC, DC, and MPEG7 and in electric commerce domain, they have standard including cXML, ebXML, Rosettanet, eCo, and so on. Table 2 shows the characteristics of each approaches including MSDL. With MSDL, conversion accuracy is relatively high,

Table 2. Comparison to Other Approaches

	1:1 Mapping	Global Schema	Standard Schema	MSDL
Conversion accuracy	Very High	High	Middle	High
Initial cost	Low	Middle	Very high	High
Maintenance cost	Very High	Middle	Low	Low
Extensibility	Low	Middle	Middle	High
Global schema	Not exist	Exist (Bottom-up)	Exist (Top-down)	Exist (Hybrid)
Application Domain	Not limited	Enterprise Integration	Specific Domain	Inter Domain

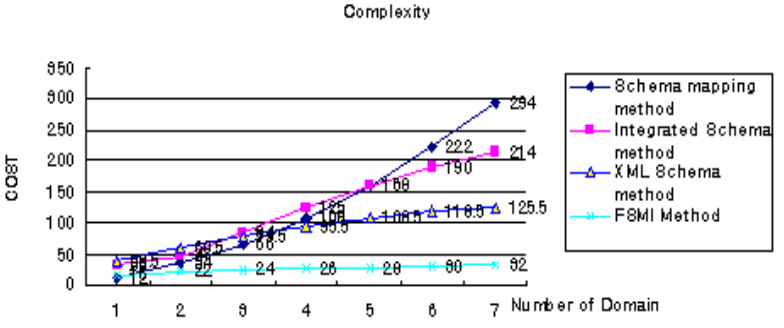


Fig. 6. Cost Graph of Complexity

because all the needed part of schema for sharing can be covered in MDR using registering process. Figure 6 shows cost evaluation graph. The cost evaluation consists of maintenance cost, center complexity, user complexity and construction cost. It shows FSMI is suitable for maintain and construct than other methods. And FSMI based on MDR is a hybrid structure system. It is easy to apply other domains system and framework. [10,11,12]

Most of all, maintenance cost in case of changing schema, is very low and extensibility in case of adding new metadata is relatively high. However, the building cost of MDR is very high and time consumed, and there may be some data losses in two steps transforming process.

6 Conclusion and Future Works

XML is a de facto standard for representing information resources in distributed environment. But it is necessary to share metadata of XML document defining syntax, structure and semantic of the document.

In this paper, we propose MSDL as a tool for describing difference between standard metadata and local XML schemas to support interoperability of XML documents. With MSDL, users can describe the syntactic, structural and semantic difference of their

XML schema to other's schema, and based on this description, can convert other system's XML document to a document with their own schema.

Users, who wish to exchange their document, just describe the difference of their XML schema with standard metadata in MDR, and make a MSDL document on the schema, and then other system can convert the document to understandable ones.

MSDL overcomes limitations of 1:1 mapping, global schema and standard schema approaches, providing low maintenance cost on schema changes, high extensibility in increase of systems, and domain independent document sharing.

In the future, we have plans to develop the converting system using MSDL, and extend it to support legacy solutions including EAI, B2Bi, Web Service, and so on.

References

1. Rik Drummond, Kay Spearman, "XML Set to Change the Face of E-Commerce", Network Computing, V.9, N.8, pp.140-144, May 1998.
2. "Chemical Markup Language", Version 1.0, Jan.1997. available at http://www.venus.co.uk/omf/cml/.
3. "Mathematical Markup Language (MathML) 1.0 Specification", W3C, July 1998. available at http://www.w3c.org /TR/1998/REC-MathML-19980407
4. ONline Information eXchange, http://editeur.org/
5. http://dublincore.org/
6. http://www.microsoft.com/biztalk/
7. http://www.ebxml.org
8. http://www.rosettanet.org/RosettaNet/
9. "Information Technology - Metadata Registries", ISO/IEC 11179 Part 1 - 6, ISO, 2003.
10. N. Guarino, C. Masolo and G. Vetere, "Ontoseek: Content-based Access to the Web", IEEE Intelligent Systems, Vol. 14, No. 3, pp. 70-80, May/June 1999
11. B. Amann and I. Fundulaki, "Integrating Ontologies and Thesauri to Build RDF Schemas" In ECDL-99: Research and Advanced Technologies for Digital Libraries, Lecture Notes in Computer Science, pages 234--253, Paris, France, September 1999. Springer-Verlag.
12. Zhan Cui, Dean Jones and Paul O'brien,"Semantic B2B Integration: Issues in Ontology-based Approaches", SIGMOD Record, Vol. 31, No. 1, March 2002.

Design of System for Multimedia Streaming Service

Chang-Soo Kim[1], Hag-Young Kim[1], Myung-Joon Kim[1], and Jae-Soo Yoo[2]

[1] Electronics and Telecommunications Research Institute,
161 Gajeong-dong, Yuseong-gu, Daejeon, 305-370, Korea
{cskim7, h0kim, joonkim}@etri.re.kr
[2] Division of Information and Communication Engineering, Chungbuk National University,
48 Gaesin-dong, Cheongju, 361-763, Korea
yjs@cbucc.chungbuk.ac.kr

Abstract. In the current life, Internet is playing a key role in transferring diverse information. As the performance of Internet has been growing, transferred information has the form of multimedia streams such as video and audio. However, in current computing and internet environment, multimedia streaming services can't be accomplished smoothly due to the various factors such as internet bandwidth, computing power and so on. In this paper, we are proposing an efficient system including hardware and software in terms of the design concept and system architecture for the multimedia streaming service. We also present the implementation result for the system.

1 Introduction

As computers and high speed networks have been popularized, Internet becomes a main means of transferring information over the world in our lives. As communication networks to the home will be upgraded to 10 ~ 100Mbps bandwidth level, high quality multimedia services such as internet broadcast, remote medical services and internet video services will be more generalized [1]. For example, a FTTH (Fibre To The Home) village is planned to be constructed in Korea. Those high quality multimedia services require a system that can handle multimedia streaming services over network fast.

In general, networking operation has high cost in computer system in that it requires several data copies from one place to another and processing steps whenever data passes from one protocol layer to another [2], [3]. In order to provide efficient multimedia streaming services over network, we need to minimize the cost of networking operation. Also, service providers should deliver multimedia contents to end users with stability and efficiency over network in real time [8], [9], [10], [11]. As service requests are increased, the traffic of a back-bone network increases, and as a result, the construction cost for the back-bone network increases. This problem should also be resolved.

In this paper, we are proposing an efficient system including hardware and software in terms of design concept and system architecture for the multimedia streaming service. We also present the result of implementation and performance evaluation for the system.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 867 – 878, 2005.
© IFIP International Federation for Information Processing 2005

2 Design Concepts and Goals

2.1 Design Concepts

The proposed system is a system specialized to internet-based multimedia streaming services with high-performance communication infrastructure. In order to provide high quality multimedia streaming services, high network bandwidth is required. Also, efficient management of infrastructure and distribution of service requests are required to reduce back-bone traffic and server loads respectively.

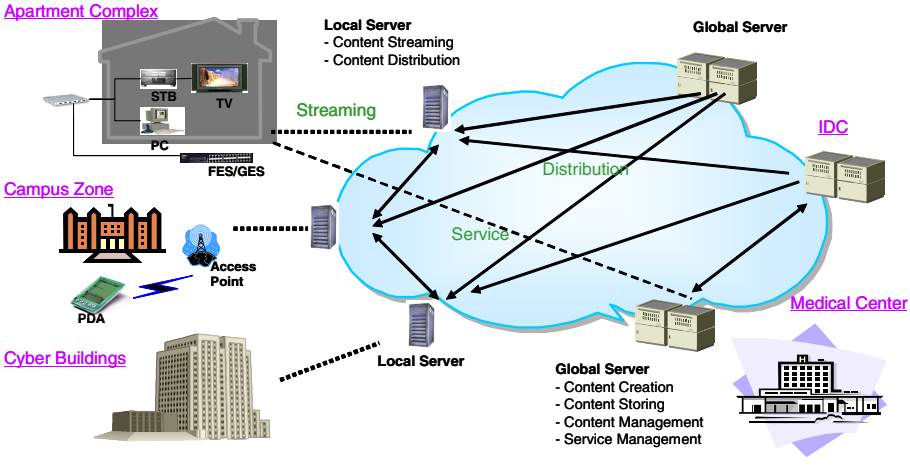


Fig. 1. Service Diagram of the System

Fig. 1 shows the service diagram of the system. Our system hierarchically distinguishes services into global and local services that are processed separately to use network bandwidth efficiently. A global server is a computer system that provides overall multimedia contents and located at Internet Data Center (IDC), e-government and so on. While a local server is a computer that provides multimedia streaming services for local communities such as apartments, companies, universities that are based on local networks. It transparently communicates with a global server. The global server should be designed to provide various functionalities for contents service and to guarantee high scalability as well as high availability. And, the local server should provide not only single-node server solution to achieve cost-effectiveness and high-performance but also multiple-node cluster server solution to build scalable and high available system.

2.2 Design Goals

As first phase, before overall system including the global server is designed, we are focusing at local server solution. Therefore, in this paper, we use the term "server" for a local server. In this case, the role of the global server as contents provider must be simulated. For this role, we will introduce CSN (Contents Server Node) node in

our system infrastructure. In order to satisfy the requirements described at section 2.1, the server has following design goals.

- The server supports up to 200 clients and provides up to 4 Gbps of concurrent streams, which is equivalent to MPEG streaming service with the bit rate of 20 Mbps.
- The server provides not only clustering solution to build scalable and high available system but also efficient content placement mechanism for distributed local servers in a cluster without shared storage.
- The server provides an efficient content distribution mechanism for storing the right content at the right time.
- The server provides an optimized multimedia file system.

In order to satisfy these goals, the server should be designed to improve performance in storage including efficient multimedia file system and networking system. Also it should support efficient clustering and content distribution features. In storage and file systems, it is needed to support continuous disk access at high speed. In networking systems, stream data stored at disks should be delivered to clients through the network in a zero-copy fashion. In clustering solution, the server should provide single system image for the cluster and efficient load balancing features between participating local servers. And, in content distribution solution, the server should provide an efficient content distribution mechanism from CSN to local servers.

2.3 Design Issues

In general, normal computer has high level of CPU utilization for networking operations with gigabit level. The reason is that complicated communication protocols are handled in host CPUs. Therefore, using a separate processor for handling networking operations in offload fashion will be useful [2], [3]. Another reason of degradation of system performance for multimedia streaming services is several data transfers and copy operations in disk-to-network path. These several copy operations are the main performance bottlenecks [3]. Therefore, several data copy avoidance architectures such as integrated layer processing [4] and direct I/O to eliminate memory-to-memory data copy are suggested trying to minimize data transfers and copy operations. We will design a special PCI card that integrates SCSI controller, TOE engine and Memory module in order to reduce CPU utilization and copy operations.

In addition, because large volume of data storage is needed for multimedia streaming services, it is also required to use striping technique such as RAID (Redundant Array of Inexpensive Disks) for fast disk reading speed [3]. The size of a streaming data is relatively large and the access pattern has continuous form. So, handling streaming data requires developing a multimedia file system that can consider the characteristics of continuous form and large size of streaming data.

According to design goals, a local server is designed to support up to 200 concurrent streams with 20 Mbps bit rate. The regional community where proposed system provides multimedia streaming services has several range of concurrent user size. From this, the server should support scalable solution. General scalable solution is clustering technique. There are several clustering related works [5], [6], [7]. The pro-

posed server has two-level cluster architecture in that the cluster consists of several server nodes and each server node can have several NS cards which will be described at section 3.1. Also it is formed like shared nothing architecture between server nodes, and in turn between NS storages of each server node. The clustering solution should support these shared nothing and two level features.

Even though the server provides large capacity of storage, all of the multimedia contents to be serviced cannot be stored at the server. As described at section 2.1, the global server plays the role as contents provider. It is important to store the right content in the right place at the right time to satisfy client's requests. Also, when a requested content is not locally available, it should be obtained from the global server or the CSN node as fast and reliably as possible. In order to support these requirements, the server should provide an efficient content distribution mechanism.

3 System Architecture

3.1 Hardware

The server consists of a conventional mother board with 2 processors and one or more special purpose PCI cards called as NS (Network-Storage) Card. The number of NS cards is ranged from one to four. The NS card is implemented as a PCI card having a PCI-to-PCI bridge, disks, a PMEM (PCI Memory), a TOE (TCP/IP Offload Engine) and two Gigabit Ethernet ports.

Fig. 2 (a) illustrates the architecture of the NS card's hardware unit. The PCI Bridge connects the local PCI bus of the NS card to the host PCI bus and transmits data directly to the network interface without disturbing host PCI bus. The Memory unit supports up to 512MB capacity of PCI memory (PMEM). The PMEM is utilized as memory space between the TOE and disks in order to support the zero-copy mechanism. The TOE is TCP/IP Offload Engine that plays the key role in offloading the TCP/IP protocol processing from the host CPU.

Fig. 2 (b) shows the software unit for the NS card, which is required for correct operations of the hardware and making its capabilities to be available to applications. The Stream Disk Array Unit supports pipelined I/O and data split mechanism, which is similar to striping technique in RAID system, to provide fast I/O of disks.

The size of a media file used in multimedia streaming services is very long. In addition, the access pattern of the media is sequential. In this case, large size of I/O unit is very helpful to provide good I/O performance. With consideration of those situations, the proposed server takes very long block size for disks. The block size can be 128K, 256K, 512K, 1M or 2Mbytes. The Linux I/O subsystem for block devices supports up to 4Kbytes block size in its kernel. This makes a conflict between the block size that NS card supports and Linux kernel can support. Because of this difference between block sizes, the server should utilize the NS character device for fast and efficient disk I/O. The SDA Block Device is also provided for compatibility of Linux. The PMEM can be used at user application directly through PMEM allocation library.

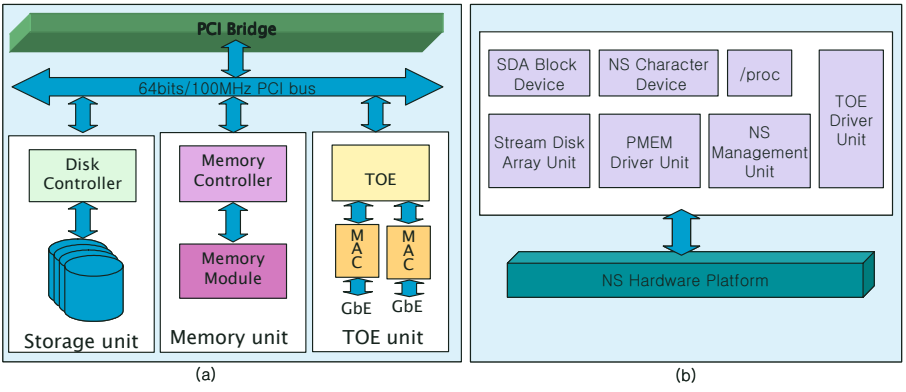


Fig. 2. Architecture of the NS Card's Hardware (a) and Software (b) unit

3.2 Multimedia File System

For high-performance multimedia streaming services, we need a multimedia file system that can utilize the facilities of the NS card and supports large volume of data, a file with large size and the access pattern of continuous form efficiently. The multimedia file system specialized to the proposed server has following features including large volume and large file size supports.

Optimized to the NS Card. The file system should handle large block sizes for high performance of disk I/O. In order to achieve this goal, the file system manipulates data through the NS character device and the PMEM**.**

Minimized Initial Access Time and Efficient File System Recovery. General file system maintains an unbalanced inode structure. This is optimized to small sized files. However, the size of streaming data is very large, and thus the unbalanced inode structure makes performance degradation in accessing multimedia streaming data having large size and cannot guarantee even initial access time. The proposed file system maintains the inode structure as a modified B+ tree structure. Also, the multimedia file system supports log-based fast recovery for maintaining consistent state of the file system at system failure.

3.3 Content Distribution

Even though the server provides large capacity of storage, all of the multimedia contents to be serviced cannot be stored at the server. Also, if services are processed globally, the traffic of the global network infrastructure increases. This increased traffic makes the service to be unreliable or unavailable. To resolve this problem, a service provider generally adopts a CDN (Content Delivery Network) technique that locates many local servers or cache servers at various specific regional areas. In a CDN infrastructure, multimedia streaming services are provided with end users through a nearest local server [8], [10], [11].

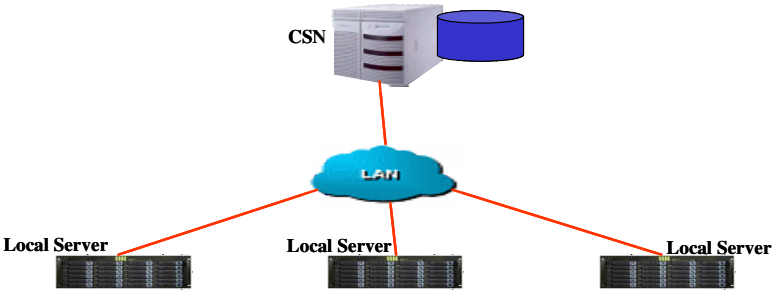


Fig. 3. Service Environment

As described at section 2.1 and 2.2, the global server plays the role as contents provider, and we introduce the CSN (Contents Server Node) node concept for replacing the global server. Fig. 3 shows the service environment for a specific regional community. With this environment, the CSN node is equipped with contents library having very large shared storage.

It is important to store the right content in the right place at the right time to satisfy client's requests. Also, when a requested content is not locally available, it should be obtained from the CSN node as fast and reliably as possible. In order to support these requirements, the server should provide an efficient content distribution mechanism. The more the server caches streaming media, the more the server can provide services to users. For this reason, our server system also adopts prefix caching mechanism for making more services to be available on the spot.

The content distribution software provides following features for versatile transfer and distribution methods, and content usage monitoring.

Content Transfer. The content distribution unit transfers contents from the CSN node to a local server. The content transfer job can be executed at promptly (On-Demand Transfer) or at delayed scheduled time (Scheduled Transfer).

Dynamic Loading. When content is requested to be serviced and if the content is not available completely, the content distribution unit transfers the content dynamically.

Preloading and Prefix Caching. The content distribution unit provides a mechanism to preload contents from the CSN prior to service requests by using either on-demand or scheduled transfer. For better chance of meeting user requests, it is desired to store as large number of contents as possible. This is a major reason of prefix caching. The content distribution unit also supports preloading with prefix size.

Content Purge. When a content transfer job is activated and if the storage has no space to store the content to be transferred, the transfer job will be failed. For preventing this situation, the content distribution unit prepares needed storage space in advance by purging some contents existed in the storage**.**

Content Storage Management. For determining the necessity of purge operation, the content distribution unit should know the status of storage**.**

Content Usage Monitoring. Selection of appropriate contents to cache at the server can affect the cache utilization and thus service performance. The unit should provide information about usage of contents**.**

3.4 Clustering Support

In order to response to user requests fast and correctly without creating unwanted delays, the server should be hosted on a cluster consisting of multiple streaming servers. Actually, the clustering technique is widely recognized to be a viable solution to build a scalable high-performance server system. The clustering unit should distribute all incoming service requests efficiently across the multiple streaming server nodes in the cluster environment. Also, it should place contents effectively across the multiple streaming server nodes for efficient and balanced resource consumption. The clustering software provides following features for efficient request dispatching and content placement.

Request Distribution for Virtual Single System Image. Even though the streaming service server is configured as a cluster, it is needed for the appearance of the streaming service server to be single system image. For single system image, the cluster has a representative virtual IP. Every client can connect to only the virtual IP, and then clustering software unit distributes all incoming service requests efficiently across the multiple streaming service nodes. In addition, each server node can have up to 4 NS cards. Since the NS card has its own IP address, pumps streaming data independently and selects shared nothing architecture, each NS card could be regarded as a distinct virtual sub-node. In other words, we should consider two-level clustering support as depicted in Fig. 4.

The request distribution called as dynamic scheduling is supported in fashion of 4-way redirection. Handling of a request in 4-way redirection fashion has following procedures as depicted in Fig. 5:

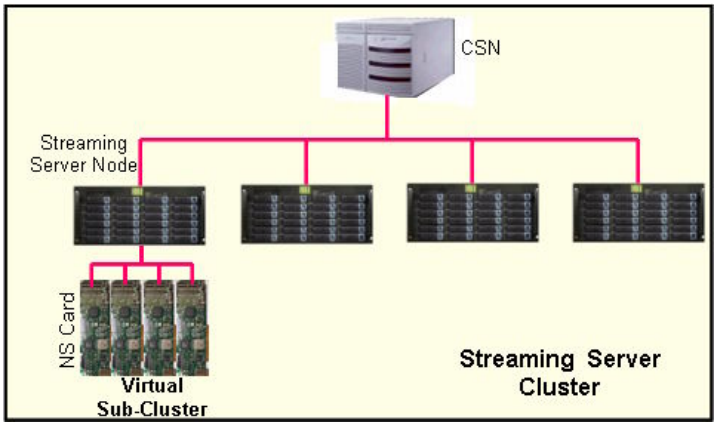


Fig. 4. Two-Level Clustering

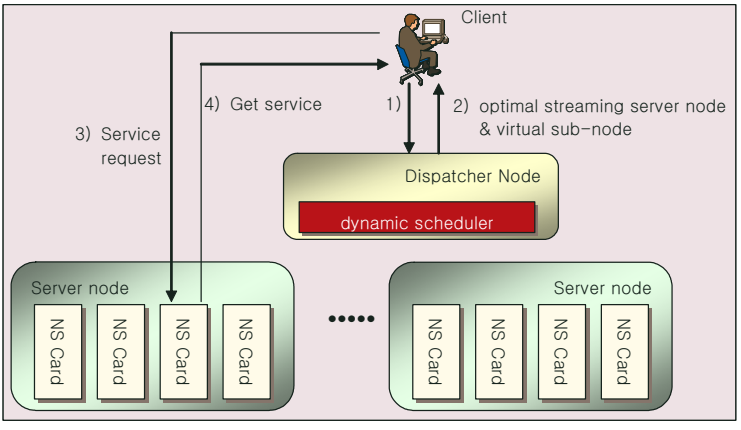


Fig. 5. Dynamic Scheduling Mechanism

1. A client requests a service at VIP
2. The dynamic scheduler at dispatcher node selects an optimal streaming server node and a NS card, and then directs the client to reconnect to the selected virtual sub-node (NS card)
3. The client reconnects to the selected virtual sub-node
4. The client is receiving the streaming service through the selected virtual sub-node within the selected streaming server node.

If a specific node or a NS card is failed, this should be removed from the request distribution target. The clustering software notifies its component's health status to prevent incorrect request distribution to failed component.

Content Placement for Efficient and Balanced Resource Consumption. In a cluster configuration, contents should be placed and managed effectively across multiple streaming server nodes and multiple NS cards in order to balance workloads. When a content is transferred from the CSN node, the clustering software should select an optimal streaming server node and a NS card to store the transferred content based on the status of each resource usage including storage. It is frequent that many service requests concentrate on the same content. In this case, if the content is located at only one NS card or one streaming server node, the NS card and the server node suffers from high overhead. The content placement mechanism should resolve this problem by replicating the content at other NS cards or streaming server nodes.

3.5 Software Architecture for Content Distribution and Clustering Support

The content distribution and clustering software architecture is shown in Fig. 6. The Job Manager which runs at a dispatcher node in the cluster, manages jobs submitted from user or other components of the system, and distributes the jobs into appropriate servers. Jobs are managed in on-demand or scheduled fashion. Also, the Job Manager handles content placement based on various information including server load, storage status, and so on.

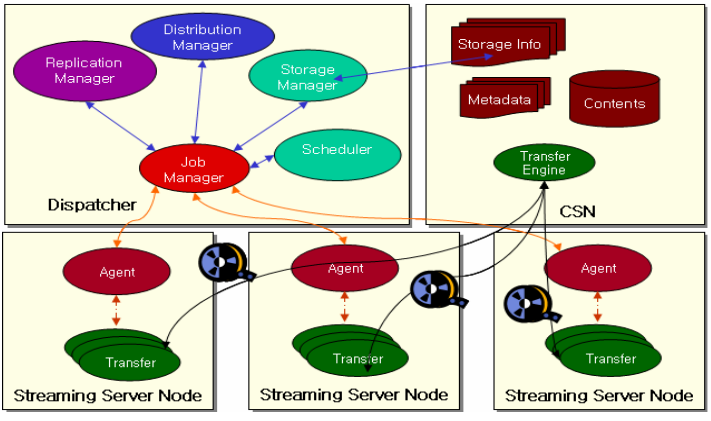


Fig. 6. SW Architecture for Content Distribution and Clustering Support

The Agent is run at every server node in the cluster, and executes jobs distributed from the Job Manager. Most of those jobs are content transfer requests. A transfer job is executed by invoking the Transfer client. The Transfer client connects to the Transfer Engine at the CSN node and pulls the requested content from the CSN. The Agent also notifies the status of streaming server node and NS cards in order to support efficient content placement and request dispatching. If the status of a component is not updated in given time period, the system considers it as failed component.

The Scheduler supports the single system image in cluster environment by dispatching all incoming service requests to optimal NS cards within streaming server nodes. It also activates dynamic loading when a requested content is not available completely. Additionally, the scheduler manages content usage information.

The Distribution Manager maintains overall metadata required to manage the content distribution, and provides utilities that control the content distribution job having on-demand, scheduled and/or prefix mode. The Storage Manager keeps track of storage usage of each NS card and if needed, activates content purging. The Replication Manager checks the necessity of replication of some contents periodically, and if needed, registers replication jobs. A replication job is processed in similar way to a transfer job.

4 Performance Evaluation

4.1 Implementation Result

The streaming server consists of a conventional mother board and a media streaming accelerator called as NS (Network Storage) card. The NS card is implemented as a PCI card having a PCI-to-PCI bridge, disks, a PMEM (PCI Memory) and a TOE. Fig. 7 shows a typical streaming server with storage box and the NS card. The server has two Intel Xeon Processors with cache size 512KB, 1.28GB Memory, on-board two gigabit Ethernet ports and up to 4 NS cards.

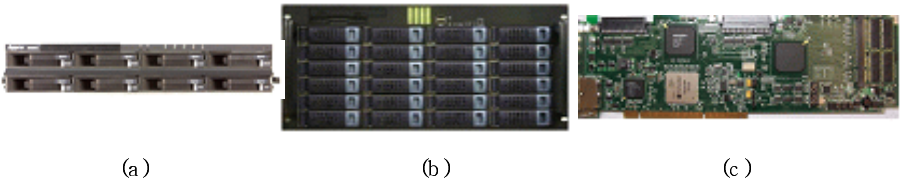


Fig. 7. Typical Streaming Server: Compact Model with 1 NS Card (a), Standard model with 3 NS Cards (b), and NS Card (c)

4.2 Performance Evaluation

We measured the performance of the proposed server by changing the type of contents format (MPEG-2, MPEG-4 and H.264), bit rate of contents and I/O block size, and so on. Also we tested the effect of clustering solution and prefix caching. All tests were run at TTA(Telecommunication Technology Association).

As shown in Fig. 8, we evaluated the capacity of stream services with zero-loss by changing the number of NS cards and bit rate on MPEG-2 contents. From the results, we can know the fact that the proposed server supports 4.1Gbps for the 4Mbps content and 4.44 Gbps for the 10M and 20Mbps contents respectively. Other test results for single-node streaming server represent similar performance.

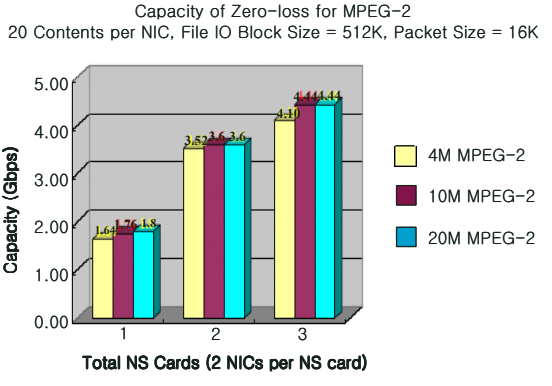


Fig. 8. Zero-loss Server Capacity for MPEG-2

Fig. 9 shows the test results related to clustering and prefix caching features. The results tell us that clustering software distributes all incoming requests evenly to the participating streaming servers. Also, we can know that prefix caching doesn't affect the streaming performance, and from which we know the fact that prefix caching can help to give more chance to hit user requests.

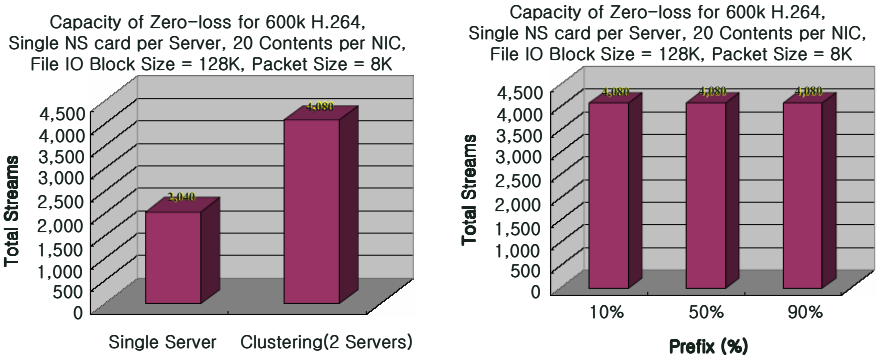


Fig. 9. Zero-loss Server Capacity for Clustering (left) and on Different Prefix Rates (right)

5 Conclusions

In this paper, we are proposing an efficient system for multimedia streaming services in terms of the design concepts, the characteristics and the system architecture including hardware and software. When designing the system, we had to consider the design goals and several design issues. The design issues for the system are the CPU utilization, zero-copy, disk throughput, multimedia file system, clustering solution and content distribution. Therefore, we used special hardware, disk striping, special multimedia file system, clustering solution and efficient content distribution mechanism.

We measured the performance on various situations and got desirable results. But, more refinement on the system for multimedia streaming services is required to enhance the overall performance and stability of system. Specially, we implemented the multimedia file system as an user-level application. The file system needs to be joined with VFS(Virtual File System) I/O sub system in Linux kernel layer. Furthermore, in order to provide total solution for multimedia streaming services, we have to design and implement the global server.

References

1. Korea National Computerization Agency, Ministry of Information and Communication: 2002 Korea Internet White Paper (2002)
2. Adaptec Company: Advantages of a TCP/IP offload ASIC. White paper
3. Thomas Plagemann, Vera Goebel, Pal Halvorsen and Otto Anshus: Operating System Support for Multimedia Systems. The computer Communication Journal, Elsevier, Vol. 23, No. 3 (2000) 267-289
4. M.B. Abbott and L.L. Peterson: Increasing network throughput by integrating protocol layers. IEEE/ACM Transactions on Networking, Vol. 1, No. 5 (1993) 600-610
5. Gregory F. Pfister: In Search of Clusters, Prentice Hall PTR (1998)
6. T. Schroeder, S. Goddard, and B. Ramamurthy: Scalable Web Server Clustering Technologies. IEEE Network, 14(3) (2000) 38-45
7. Wensong Zhang: Linux Virtual Server for Scalable Network Services. In Proc. of the Ottawa Linux Symposium (2000)

8. Kasenna: Video Content Distribution: A Hybrid DCS/Caching Architecture for Broadband Video. White paper
9. Subhabrata Sen, Jennifer Rexford, and Donald F. Towsley: Proxy Prefix Caching for Multimedia Streams. In Proc. of INFOCOM (1999) 1310-1319
10. Stardust: Content Networking and Edge Services: Leveraging the Internet for Profit. White paper
11. HCL Technologies: Content Delivery Networks: An Architectural Overview. White paper

A Workflow Language Based on Structural Context Model for Ubiquitous Computing

Joohyun Han, Yongyun Cho, and Jaeyoung Choi

School of Computing, Soongsil University,
1-1 Sangdo-dong, Dongjak-gu, Seoul 156–743, Korea
{jhhan, yycho}@ss.ssu.ac.kr, choi@ssu.ac.kr

Abstract. Workflows used in business processes and distributed computing environments have supported service automation by connecting many tasks with rules and/or orderings. To adapt these workflows to ubiquitous computing, we must specify the context information on their transition conditions. In this paper, we propose uWDL, Ubiquitous Workflow Description Language, to specify context information on the transition constraints of a workflow in order to support adaptive services, and we designed a structural context model to express the context information in uWDL. Furthermore, uWDL is designed based on Web services, which are standardized and independent of various heterogeneous platforms, protocols, and languages. In order to verify the effectiveness of uWDL, we designed and implemented a scenario described with uWDL, and demonstrated that the uWDL system provides users with autonomic services in ubiquitous computing environments.

1 Introduction

Ubiquitous is a Latin word which means that the computing environments are absorbed into the physical world and integrated in everyday life [1]. In such ubiquitous environments, more sensibility is required than in traditional computing environments. Deriving context information such as location, status and actions of users, devices, and network by sensing physical environments is necessary to provide users with context-aware services based on this context information.

The workflow model in [2] supports service automation through a sequence of rules in processing tasks. It has been successively applied to traditional computing environments such as business processes and distributed computing in order to perform service composition, flow management, parallel execution, and time-driven services. It is also the task of the workflow to provide a service to the right person or the right application at the right time so that the service for a specific task can be carried out [3]. Users of ubiquitous computing environments may want to receive services in an appropriate form, at the appropriate time, and without user intervention, on dynamically changing environments [4,5]. In order to support automatic service in these environments, we need to adapt workflows to ubiquitous computing. However, workflows must satisfy at least the following two requirements. First, a context-aware state transition is required

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 879–889, 2005.
© IFIP International Federation for Information Processing 2005

to provide users with appropriate services according to a user's current situation. Second, a platform and language-independent standard service interface needs to integrate, manage, and execute ubiquitous applications which consist of heterogeneous protocols on various platforms [6].

In this paper, we propose uWDL (Ubiquitous Workflow Description Language) which specifies the context information on the transition conditions of workflow services to provide users with an adaptive service for a user's current situation. It specifies context information as the perspective of rule-based reasoning which can effectively infer his/her current situation in a simple and flexible way. We also designed a structural context model to express the context information in uWDL. Furthermore, uWDL is designed based on Web services interfaces, which are not only already specified and widely used but independent of various platforms, protocols, and languages. By interpreting a scenario described with uWDL and executing the scenario, the uWDL system can effectively provide users with context-aware and autonomic services.

2 Related Work

UBL(Universal Business Language) [7], CC(Core Component), and BIE (Business Information Entity) [8] provide a formalized document style based on the ebXML [9] framework to companies for systematically supporting interoperability of the documents across the companies. These components are used in e-Business for designing the CC with UML(Unified Modeling Langauge), adapting the business context to BIE and translating the BIE into various form such as XML schema and DTD for machine readability. However, because they are designed for the purpose of e-Business, it is difficult to efficiently specify context information dynamically obtained from the sensor network in ubiquitous computing environments. uWDL can specify context information aggregated from the various sources in ubiquitous computing. As it is a workflow language, it can easily represent a flow of adequate services and select the service according to the user's situation. Furthermore, because uWDL is designed to support Web services which are platform and language-independent standard service interfaces, it can express dependency and parallel execution among the services in the heterogeneous ubiquitous computing environments.

BPEL4WS [10], WSFL [11], and XLANG [12] are Web service-based workflow languages for business processes and distributed computing environments. They support service transition, and use XML-typed messages defined in other services using XPath. However, context information is a complex data set that includes data types, values, and relations among the data types. XPath cannot sufficiently describe such diverse context information as this because it can use only condition and relation operators to decide transition conditions. uWDL uses a context triplet - subject, verb, and object - in order to express high-level context information as transition conditions which existing workflow languages are unable to support.

3 A Structural Context Model

A context in ubiquitous computing environments can indicate any information which is used to characterize the situation of an entity [13]. A context-aware application uses context information and performs context-appropriate operations [5,13]. In ubiquitous environments, all services head for context-aware services to provide services appropriate for the user's situation. In order to provide a context-aware workflow service in ubiquitous environments, an appropriate service is selected and executed based on the context information. Workflow services must be executed conditionally, concurrently, or repeatedly according to the conditions specified in the workflow. It is also required that the workflow specifies the constraints that influence the execution order of services such as start time, end time, deadline, and conditions.

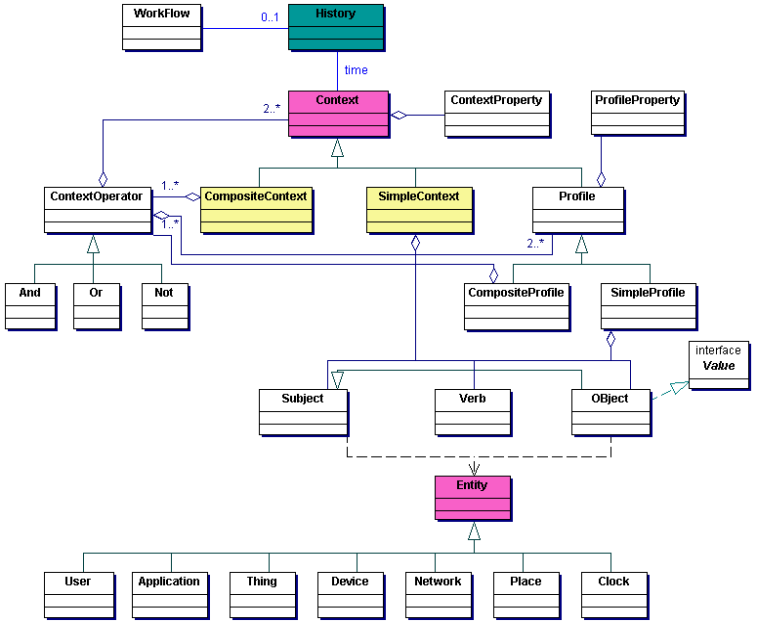


Fig. 1. The structural context model

Figure 1 shows the class diagram of the structural context model. The structural context model expresses ubiquitous context information from the viewpoint of knowledge structure. Because it has an information structure to express complex context information, it is possible to describe contexts in order to define the constraints of the state transition in uWDL. The following subsections explain some components which are context, entity, context triplet, simple context, and composite context shown in Figure 1 to specify the context information on the transition condition of services in uWDL scenario documents.

3.1 Context and Entity

The context used in ubiquitous computing is very different and varies according to domains. Generally, contexts are classified by categories such as time, location, identity, and activity. Any information obtained from ubiquitous middleware is objectified with an entity. An entity represents a person, a place or an object relevant to the interaction between a human and the corresponding virtual computer world. A context means information used to characterize the situation of entities. An entity categorized into a specific domain has the type of the domain and the value of its type.

In ubiquitous computing environments, the context information is expressed as entities and the entities are decided by the domains. For example, every entity, such as a human, application, goods, device, network, location, and system timer, constitutes all environments in human life [13]. Context information, such as the location and behavior of a man, device status information, an object in a nearby location, schedule information, an application running on a computing device, the network bandwidth, temperature, brightness, and volumn of noise, is obtained by these entities. Such context information is obtained by sensing the physical environment's entity or profile information. Profile information is the explicit information of entities. Context information obtained by sensing the environment varies dynamically over time, but profile information is fixed or varies slowly [14].

3.2 Context Triplet

In Figure 1, a context can be composed of several independent entities obtained from a sensor network. Context has a sequence of subject, verb, and object according to the meaning of the entities. This sequence is called a context triplet. The context triplet is similar to the concept of RDF(Resource Description Framework) [15]. RDF is a language to describe a resource's meta-data and expresses a resource as a triplet of {subject, predicate, object}. Because RDF can efficiently describe a knowledge structure, most ontology languages are based on RDF.

A context can be described through a context triplet associated with a particular situation. For example, let's suppose a situation, "John studies in room 301". We can divide the situation into a context triplet which is a {(PersonType, John), (ActivityType, study), (RoomType, 301)}. Each of the entities corresponds to subject, verb, and object in the context triplet according to their meanings, and in some cases a context would be composed of more than one context triplet for describing the specific situation. Finally, the context expressed by the context triplet is compared with the entities obtained from a sensor network for service transition in uWDL.

3.3 Simple Context and Composite Context

Context information varies from a low level context describing facts in physical environments to a high level context obtained by artificial intelligence. In Figure 1, context and profile information is described using a context triplet to express

situation information. This context triplet is called simple context information. Some context information cannot be expressed only in simple context information. Such context information is called composite context, which combines a set of simple context information using the 'and', 'or', and 'not' operators.

4 A Ubiquitous Workflow Description Language

Although current workflow languages such as BPEL4WS, WSFL, and XLANG can specify the data flow among services based on Web services, these workflow languages do not support the ability to select services using context, profile, or event information in ubiquitous computing environments. Therefore, it is difficult to express relationships among the services using traditional workflow languages in ubiquitous environments.

uWDL (Ubiquitous Workflow Description Language) is a Web service-based workflow language that describes service flows and provides the functionalities to select an appropriate service based on high-level contexts, profiles, and events information, which are obtained from various sources and structured by Ontology [16]. To provide these functionalities, uWDL specifies the context and profile information as a triplet of {subject, verb, object} based on a structural context model for rule-based reasoning which can effectively represent the situation in a simple and flexible way. Figure 2 shows the schema structure of uWDL.

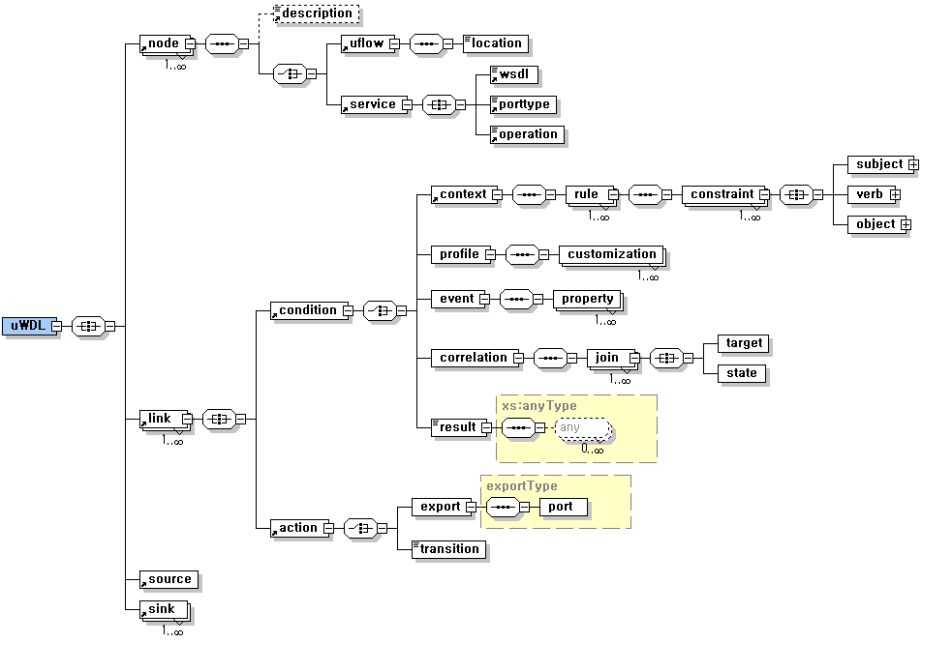


Fig. 2. uWDL schema

Table 1. EBNF Notation of the <node> Element

Node	::= Description?, (Uflow | Service)
Uflow	::= Location
Service	::= Wsdl, Porttype, Operation

4.1 <node> Element

The <node> element points to an operation that provides a functionality of Web services in ubiquitous environments. Web services use WSDL (Web Services Definition Language) to describe the port types and operations of specific Web services. Therefore, uWDL uses the <service> element and subelements of the <service> - <wsdl>, <porttype>, and <operation> - to describe the service location, type, and operation of a specific Web service, and the <uflow> element directs a reference to another uWDL document. Table 1 shows the EBNF(Extended Backus Naur Form) notation of the <node> element.

4.2 <link> Element

The <link> element is the most important part of uWDL. It specifies context, profile, and event aggregated from ubiquitous environments and defines the flow of services. The <link> element is composed of <condition> and <action> elements. The <condition> element uses <context>, <profile>, and <event> subelements to specify the context, profile, and event status of a specific node, respectively. If the calculated value of the status satisfies a given condition, the action described in the <action> element is performed. The <action> element consists of <export> and <transition> elements, where <export> has a control link and a data link according to its attribute, and <transition> specifies the state change of the current node.

The <condition> element is responsible for selecting the appropriate service based on the context, profile, and event information. The important element is <context>, which contains <constraint> to specify context information standardized by Ontology. The <constraint> element has the subelements of <subject>, <verb>, and <object>, and is designed based on the context triplet of the structural context model in Section 3. The information of subject, verb, and object represents abstract information such as location, computing device, user activity, and the social situation of the diverse domain in ubiquitous computing environments. The <constraint> element expresses a context based on the relationship of the subject and the object, which are instances of the entities. The <subject>, <verb>, and <object> elements also have an attribute of 'type'. The type attribute represents a property of an entity in a domain. The composite attribute of the <constraint> element has an attribute value of 'and', 'or', and 'not'. By using these values of the attribute, it is possible to express

Table 2. EBNF Notation of the <link> Element

Link	::= Condition, Action
Condition	::= Context | Profile | Event | Correlation | Result
Context	::= Rule+
Rule	::= Constraint+
Constraint	::= Subject, Verb, Object
Profile	::= Customization+
Event	::= Property+
Correlation	::= Join+
Join	::= Target, State
Action	::= Export | Transition
Export	::= Port

the relationship among the simple contexts and describe a high-level complex context. The <rule> element means a set of the <constraint> elements, and represents the high-level expression to infer a social situation. Table 2 shows the EBNF(Extended Backus Naur Form) notation of the <link> element.

5 The Architecture for Handling Contexts in uWDL

A uWDL document designed for a specific scenario must be translated and executed to provide an adaptive service for a user's situation. To achieve this purpose, we need a process to manipulate contexts aggregated from a sensor network. Figure 3 shows the architecture for handling the contexts expressed in uWDL. The uWDL parser parses an uWDL scenario document and produces a DIAST (Document Instance Abstract Syntax Tree) [17,18] as a result. A DIAST represents the syntax of a scenario document. A DIAST is used to compare contexts expressed in a scenario with entities aggregated from a sensor network to verify their coincidence. A context consists of a triplet of {subject, verb, and object} in sequence. A context is described with one or more constraint elements, and each constraint is represented by a triplet. These triplets are used as nodes to construct the DIAST's subtree. In Figure 3, the partial subtree in dotted lines indicates a subtree that makes up context constraints in the scenario.

The context mapper extracts types and values from objectified entities aggregated from the sensor network, and composes a subtree which consists of subject, verb, and object information. The context mapper then compares the type and the value of an entity with those of the constraint elements in the DIAST's subtree. If the type in the entity matches with its counterpart in the constraint element, the context mapper regards it as a correct subelement of the constraint element. If each entity has the same type, it may be ambiguous to decide a context's constraint according to its entity type only. The problem can be resolved by comparing the value of the objectified entity with that of the constraint element in the DIAST's subtree.

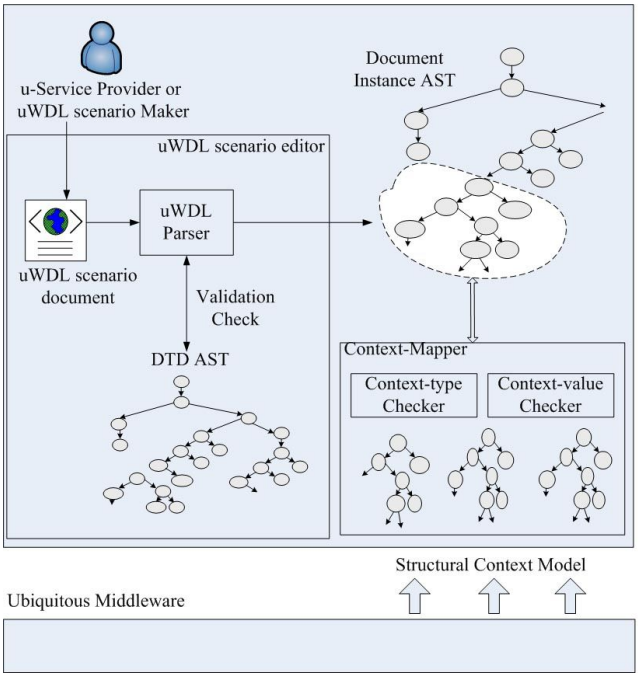


Fig. 3. The architecture for handling the context in uWDL

6 Experiments

In this section, we show a process to decide the state transition according to the context information. The purpose is "Implementing a service which prepares an office meeting automatically according to a user's schedule." The scenario designed by using the uWDL scenario editor is as follows: "John records an appointment on his notebook computer that there is a presentation in Room 313 at 10:00 AM. John moved to Room 313 to participate in the meeting at 9:40 AM. There is a RFID sensor above room 313's door, and John's basic context information (such as name and notebook's IP address) is transmitted to a server. If the conditions, such as user location, situation, and current time, are satisfied, then the server automatically downloads his presentation file and executes a presentation program."

For testing, we developed a uWDL scenario editor with which we created a scenario of an office meeting. Figure 4 shows the uWDL scenario editor. The uWDL scenario editor is composed of available services, constraint information, and so forth. The available services show a list of the Web services available in the current environment. The structure window shows the structure of the DIAST's subtree for the constraint element highlighted in the editing window.

Figure 5 shows a uWDL instance document created with the uWDL scenario editor for the above scenario. The constraint information presents a list of the

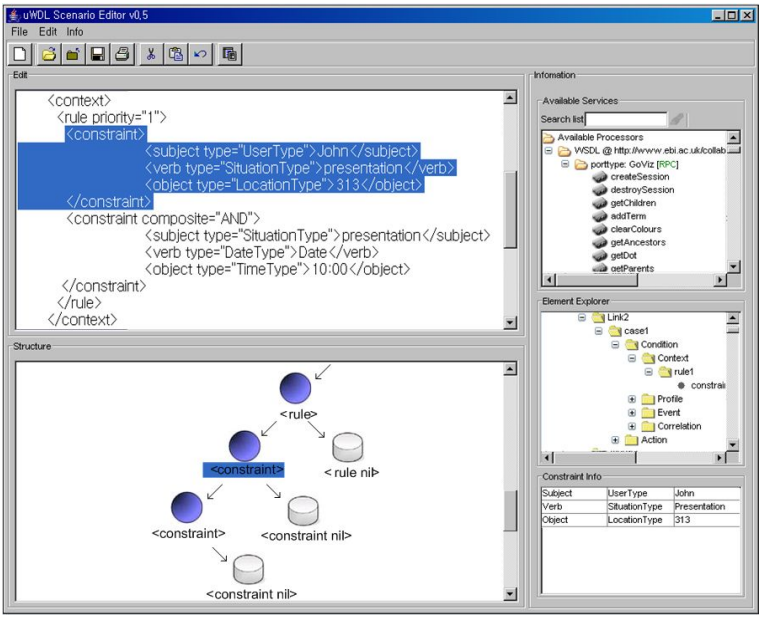


Fig. 4. uWDL scenario editor

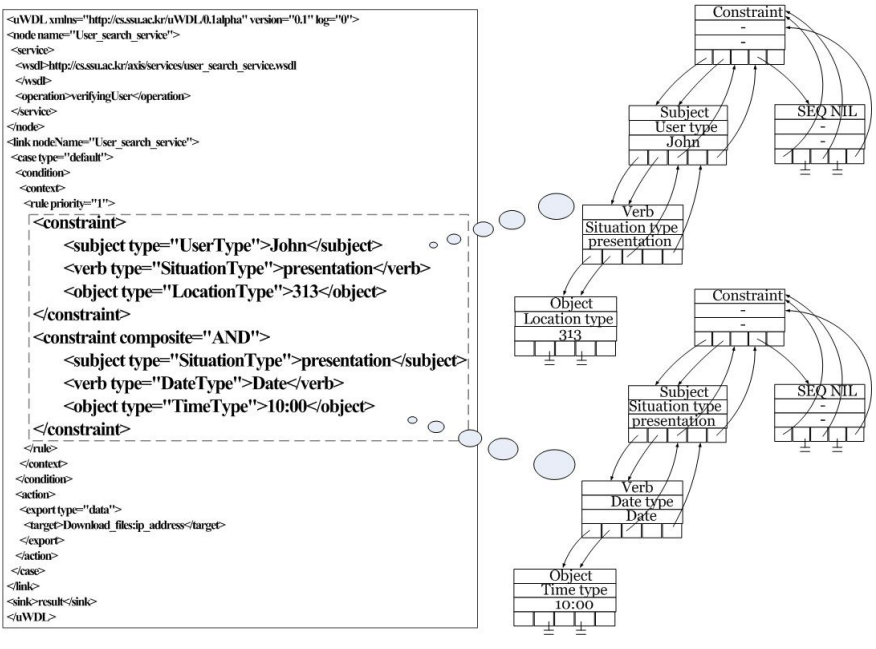


Fig. 5. The office meeting scenario and the DIAST's subtree produced by the uWDL parser for the scenario

entity types aggregated from a sensor network within the current environment, and a list of the entity values registered in each entity type. If the context mapper receives entities objectified as (SituationType, presentation), (UserType, Michael), (UserType, John), and (LocationType, 313) from the sensor network during processing of the scenario, it compares the entities' types and values with those of the constraint element highlighted in the DIAST's subtree of Figure 4. At the moment, because the entity (UserType, Michael) is not suitable for any one of the subtree's elements, it is removed. As a result, the entity (UserType, John) is selected as a context for the DIAST's subtree.

7 Conclusion

In this paper, we proposed uWDL (Ubiquitous Workflow Description Language) which can describe service flows for a ubiquitous computing environment, and we designed the structural context model to express the context information in uWDL. Because uWDL is based on Web services, it is able to integrate, manage, and execute various heterogeneous services in ubiquitous environments. In a ubiquitous computing environment, a service engine needs a method to recognize a user's context and situation information to decide the state transitions of a service flow. As explained in Section 2, current workflow languages are for business processes and distributed computational workflows, and they are not suitable for describing the context and situation information needed for service transitions in a service flow of ubiquitous computing environments.

We designed uWDL so that it can specify the context information on transition constraints of a service workflow in ubiquitous computing environments. As a result, users are provided with an appropriate service according to the user's context information. We implemented an uWDL scenario editor and developed a sample scenario described with the uWDL, and we demonstrated that the uWDL system provides users with autonomic services in ubiquitous computing environments. In the near future, we will expand the uWDL schema to express more detailed situations by assigning semantic information to Web services.

Acknowledgements

This research is supported by the Ubiquitous Autonomic Computing and Network Project, the Ministry of Information and Communication (MIC) 21st Century Frontier R&D Program in Korea.

References

1. M., Weiser: The Computer for the 21st Century. Sci. Amer. (1991)
2. D., Hollingsworth: The Workflow Reference Model. Technical Report. TC00–1003. Workflow Management Coalition (1994)
3. Wil, van, der, Aalst, Kees, van, Hee: Workflow Management, Models, Methods, and Systems. The MIT Press. pp.147 (2002)

4. D., Saha, A., Mukherjee: Pervasive Computing: A Paradigm for the 21st Century. IEEE Computer. IEEE Computer Society Press (2003) 25–31
5. Guanling, Chen, David, Kotz: A Survey of Context-Aware Mobile Computing Research. Technical Report. TR200381. Dartmouth College (2000)
6. Mack, Hendricks, Ben, Galbraith, Romin, Irani, et al.: Professional Java Web Services. WROX Press. (2002) 1–16
7. Bil Meadows, Lisa Seaburg, "Universal Business Language 1.0", OASIS Committee Draft, Sep. 2004.
8. "Core Components Technical Specification V2.01", UN/CEFACT Technical Specification, Nov. 2003.
9. Anders Grangard, Brian Eisenberg, Duane Nickull, "ebXML Technical Architecture Specification v1.0.4", OASIS, Feb. 2001.
10. Tony, Andrews, Francisco, Curbera, et al.: Business Process Execution Language for Web Services. BEA Systems. Microsoft Corp. IBM Corp., Version 1.1 (2003)
11. Frank, Leymann: Web Services Flow Language (WSFL 1.0). IBM (2001)
12. Satish, Thatte: XLANG Web Services for Business Process Design. Microsoft Corp. (2001)
13. Anind, k., Dey: Understanding and Using Context, Personal and Ubiquitous Computing. Vol 5. Issue 1. (2001)
14. Karen Henricksen, Jadwiga Indulska, Andry Rakotonirainy: Modeling Context Information in Pervasive Computing Systems, Pervasive 2002, LNCS 2412 pp.167–180 (2002)
15. W3C: RDF/XML Syntax Specification, W3C Recommendation (2004)
16. Deborah, L., McGuinness, Frank, van, Harmelen, (eds.): OWL Web Ontology Language Overview. W3C Recommendation (2004)
17. Aho, A., V., Sethi, R., Ullman, J., D.: Compilers: Principles, Techniques and Tools. Addison-Wesley (1986)
18. Bates, J., Lavie, A.: Recognizing Substring of LR(K) Languages in Linear Time. ACM TOPLAS. Vol.16. No.3. pp.1051–1077 (1994)

Ubiquitous Content Formulations for Real-Time Information Communications

K.L. Eddie Law[1] and Sunny So[2]

[1] Ryerson University, Electrical and Computer Engineering,
Toronto, Ontario, Canada M2K 2Y2
eddie@ee.ryerson.ca
[2] ViXS Systems, Inc., Toronto, Ontario, Canada M2J 5B5
sunny.so@utoronto.ca

Abstract. With rapid advancements in wireless devices, ubiquitous computing seems becoming a reality everyday. Active pervasive network infrastructure has been introduced to offer selective and intelligent information communications according to access bandwidths of end users' devices. In this paper, research has been extended to deal with mixture of critical and non-critical information. Data classifications are devised for the infrastructure to interpret the importance of data accurately. The design relieves computation requirements at end devices, and mediates delivered information based on users' personal preferences. Further, the operations of content adaptations are transparent to all end devices, and users always perceive high quality transactions and are satisfied with offered network services. More importantly, the resulting design further improves overall system throughput and delay performance.

1 Introduction

Portable wireless devices are widely deployed and used nowadays. More and more people are actively involved in communications while traveling. The ubiquitous computing[1] paradigm [1] appears to be a reality sooner everyday. Weiser [1] indicated that "the most profound technologies are those that disappear." In pervasive computing [1]-[3], information should permeate through networks to seamlessly reach an end user with the highest degree of embeddedness and mobility. When a person moves, his or her portable device can encounter wireless access links with fluctuating communicating bandwidths. Definitely, user always wants all active connections remain connected through the Internet without any interruption due to the mobility. Further, the size of a portable device is usually small, and has limited power supply with constrained computational resources. Hence, it is likely too demanding to add the function of content adaptation into a portable device. We should reduce the amount of computations at a device for extending the operating duration of a battery. Besides, the quality of a wireless

[1] Ubiquitous computing and pervasive computing are used interchangeably in the paper.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 890–899, 2005.
© IFIP International Federation for Information Processing 2005

network connection changes frequently, dramatically, and without warning. The embeddedness feature (e.g., the connecting interface) of a mobile device may have altered (e.g., from wireless LAN to cellular network) without informing the user. Applications must cope with these changes automatically. Appropriate designs should be carried out to make a pervasive computing system context-aware. It is always a challenge to identify a reference operating platform upon which disparate potential solutions are coalescing for pervasive computing.

Active pervasive network infrastructure (APNI) [4] has been proposed to offer proactive transparent operations for pervasive computing. The infrastructure handles abruptly changing network conditions instantly based on users' preferences, and strives to retain the integrity of information by adapting the content using available computing resources on overlays or in networks. The network infrastructure should be augmented with components and algorithms for providing high-performance pervasive computing, and the goal is to provide desired interpretable content information at destinations with the highest accuracy.

Accuracy of content adaptation should be based on appropriate classifications of information. In the paper, operations on classified data on per user's basis will be examined. The criteria on executing transparent real-time adaptation are to make service subscribers perceive high-speed data transmissions, high quality transactions, and are satisfied with the services provided by the service and content providers. Thorough analysis on achieving these criteria will be reported.

2 Transaction Procedures and Data Classifications

A brief review of the active pervasive network infrastructure (APNI) [4] is discussed. The system architecture is shown in Fig. 1. It can be constucted using agent technology, or programmable node (active networks) [6, 7]). Programmable agents at the edges of networks and programmable nodes inside networks virtualize the interactions between any end users' devices and content providers. The design objective is to mediate delivered information according to a user's personal preferences and its dynamic content adaptation operations are totally transparent to both the content providers and mobile devices.

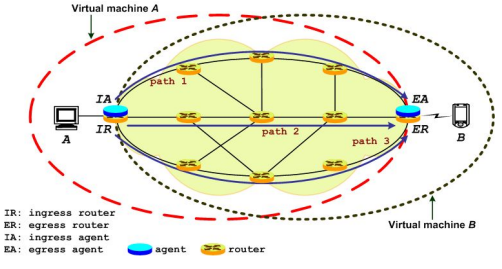


Fig. 1. Active pervasive network infrastructure

Agent technology constitutes an important part of the pervasive network infrastructure. All end users' computing devices connect to the Internet through access routers where the agents always locate. For an active communication session, a sender connects to a router with an ingress agent, whereas the egress agent is at the router that connects to the receiver. Indeed, an access router always contains both the ingress and egress agents for monitoring all connections in and out the Internet. At these points of attachments, the egress agents monitor the quality changes of devices' connecting interfaces; while the ingress agents administrate the operations inside the network infrastructure for appropriate content adaptations.

2.1 Desirable Client/Server Transactional Model

Data transactions are usually transported using Transmission Control Protocol (TCP) [8] sessions. The associated three-way handshake procedures and sequence number acknowledgment operations at the transport layer offer the reliability that we are seeking for on data communications. The slicing window mechanism allows proper amount of data delivery under different network conditions. However, when it comes to user's application, the data encoded and decoded at the application layer may or may not be able to change. For example, images can be re-compressed but not one bit of critical data can be removed. As a result, application layer notification should be sent, for example, through a new request update message as described in Fig. 2(a).

Indeed, a client usually provides personal preferences when paying for and subscribing a service content from a content provider. The preferences can be

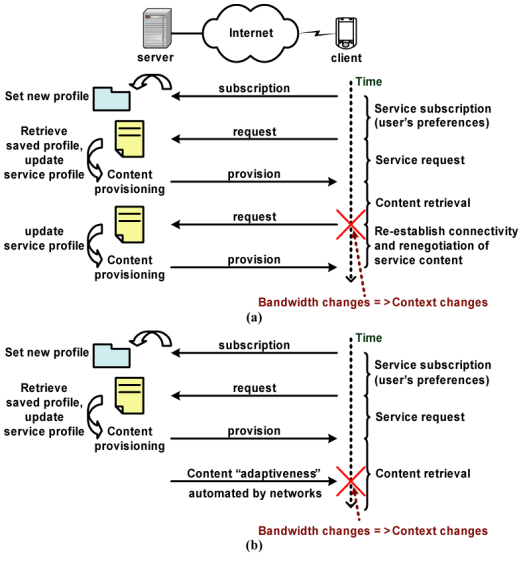


Fig. 2. Service subscription: (a) regular updates, (b) desirable mechanism

considered the quality of service (QoS) requirements of a user at the application layer. Then when the client sends a request message to retrieve subscribed information, the provider should deliver content information according to subscriber's contract. Unfortunately, complete information delivery may be forbidden due to the latest network condition. This implies only context change can possibly complete the expected transaction. One possible method is to design a renegotiation process [5] for changing system parameters according to the new network conditions. Unfortunately, this procedure may not run automatically. Furthermore, a client may sometimes need to have certain technical knowledge to pass the new settings to the provider. Therefore, the operations may not be user-friendly and may be time consuming.

As a result, we propose a transactional model at the application layer for the pervasive network infrastructure. Fundamentally, the three-step operational procedures (i.e., the subscription, request, and provision stages) should be kept in application transactions. It is the coordinations of the peer agents for carrying out the "adaptiveness" features in the proposed framework as shown in Fig. 2(b). However, how can the pervasive network infrastructure know what to change in context?

2.2 Content Description Protocol

Context-awareness adaptation can be assisted with content classifications. However, it is always the variations of network resources that lead to the needs of content adaptations. Actually, the bottleneck links are usually the wireless downstream links connecting from egress routers to portable devices. These devices are owned by service and content subscribers. A user sends a short request message to obtain the subscribed information, that may consist of a large chuck of data. Furthermore, the requested information may possibly consist of multiple connections. For further advancement in future network transaction operations, it is desirable to introduce the concepts of content classifications with a transactional operating model.

In the protocol design, when a user subscribes certain services, personal preferences should be posted to content providers. An appropriate content description protocol may need to be designed to take advantages of the design. A three-step procedure is shown in Fig. 2(b). For example, a subscription message is sent to a TCP port number (e.g., port XYZ) defined at provider's server. The packet format should be standardized or well-defined for any content and service providers to undergo content adaptation for pervasive computing. The egress agent on a router carries out two functions: 1) intercept, and 2) interpret the request message. The functions of the subscription, request and provision messages in Fig. 2(b) are outlined with basic fields depicted in Table 1. Whenever a subscription packet goes to port XYZ is received at an edge router, the agent intercepts the message. Firstly, the egress agent appends the bottleneck link's parameters, e.g., the available residual bandwidth and measured/estimated delay. The appending operation is to inform the ingress agent connecting to the service provider to administrate proper content adaptations inside network infrastructure. Con-

Table 1. Transaction protocol messages

Message name	Function and Fields	
Subscription	Carries subscriber's preferences	
	service port number	service provider's defined port number for the service
	content number	service provider's defined content number
	content nature	real-time critical data
		real-time non-critical data
		non-critical data
		images (are preferably kept)
		images (can be removed)
	service type	must receive
		preferably receive
		can receive
		can drop first
	(more 'content number, content nature, service type' fields)	
Request	A user initiates a service request to provider's port number	
Provision	Subscribed service content is being delivered in "provision" packets	
	'content number, content nature, service type'	

ceptually, the two agents are negotiating QoS settings for the application when content adaptation is needed. It operates differently from traditional end-to-end adaptation schemes. If active network paradigm is used, the active routers between the two agents update the constraint parameters, if appropriate. When the subscription message arrives at the server's ingress router, the ingress agent removes the appended network constraint information, and delivers the original subscriber's subscription message to provider.

The ingress agent should have registered the network costraint parameters and user's content preferences. Subsequently, it can interpret a subscriber's subscription contents for administering the operations in infrastructure to distribute pervasive computations for the receivers. Typically, an ingress agent do not undergo any pervasive computations to avoid itself becoming the computation bottleneck in networks. Hence in the proposed model, messages are intercepted, examined, and modified, if needed, by agents in accordance to the client context, network service availability, user preferences, and network QoS situations. This eliminates the need for a client to be aware of the complexity of networks, thereby maintaining a server virtual machine model. On the other hand, the request and provision messages are simplified since QoS services at the application layer and content adaptations are provided by transit servers or routers (through the delegation from the agents) in lieu of the server. This hides the requirements from the clients, thereby achieving a client virtual machine model.

2.3 Data Classifications and User's Preferences

Content in transit being adapted to access context and available resources should have the involvements of end users' decisions. Otherwise, the adaptation process cannot provide the highest perceptual and perceived values to end users. Human, physical and cognitive environmental factors affect the tailoring of a computational model to offer ubiquity, embeddedness, and adaptiveness. Some setup parameters are shown in Table 1. In this section, more thorough discussion on

designing user's preferences should be made for implementation in active pervasive network infrastructure. Basically, we should consider the

- real-time constraint:
  - if yes, then its expected value should be set;
- criticality of information:
  - if yes, then function of reliability should be enabled;
- ranking of information:
  - $R_1 > R_2 > \cdots$ (four ranking classes in testbed, see Table 1),
  - R_1 belongs to the "must receive" class,
  - may be ranked by providers to avoid disputes;
- allowed cognitive distortion of images, voices and video:
  - if accepted, then the rate of distortion should be indicated;
- preferred presentation language:
  - for example, English, French, German etc.

2.4 Content Adaptation for Real-Time Delivery

Mobile users always expect information being retrieved to arrive briefly after they send the service requests. For example, many servicemen rely on real-time service tickets to determine the next service locations. Thus, real-time communications are more desirable for certain mobile users. Therefore, the pervasive computing system should provide an expedited delivery service as an important service feature to end users. In this design, a user marks down W as his or her limit of patience on waiting time. Or it can be considered as the maximal time limit within which the content allow a user to make reasonable interpretation, and it may have an acceptable perceptual value when received. The W is called expected real-time constraint.

When a path is given, a client needs to send a request to retrieve subscribed information. Suppose that the total round-trip delay between client and server is T. Although it should rarely happen that the expected real-time constraint of a client is smaller than the round-trip delay, i.e., $W < T$, the server service can never satisfy a client's expectation traditionally. But with the content adaptation mechanism, the size of receiving information can be changed, supposingly, based on the ranked information in data. Consequently, it may possibly satisfy the W requirement. In the following, let assume the receiving data can be truncated or compressed for developing an algorithm. In ubiquitous computing, the last mile (i.e., the wireless access link) is usually the slowest link in a connection. The size of packet should then be adapted to go through the bottleneck link and the resulting size is denoted as s_A.

Since high-speed connection is usually arranged for high-performance server system, the propagation and transmission delays between server and ingress agent, and the processing and queueing delays at server are assumed to be negligible. Besides, the processing and queueing delays at client are ignored while the downlink delay from the egress agent to client may likely be the bottleneck parameter. Suppose $t(\cdot)$, $p_p(\cdot)$, $p_c(\cdot)$, and $q(\cdot)$ are the transmission delay, propagation delay, processing delay, and queueing delay, respectively.

Further, the $t(ER)$ denotes the transmission delay of a packet from egress agent to receiver, i.e., $t(ER) = \frac{s_i}{B}$ where s_i is the size of packet i and B is the downlink bandwidth of access channel. Similarly, IE and SI indicate the values of a variable from the ingress to egress agents, and from sender to the ingress agent, respectively. The notations are swapped in the reverse direction.

There may have multiple paths between any two agents. But when a path is given, the associated $p_p(\cdot)$ is fixed. Furthermore, if the size, s_i, is unchanged, then the $t(\cdot)$ is also fixed. However, the sizes of packets may get modified and adapted to constrained network resources. Consequently, $\sum t(\cdot)$ can be a varying parameter. Similarly, the programmable nodes may have different processing times for packets with different data and program code. Then, the parameters, s_i and $\sum p_c(\cdot)$, also vary.

Round-trip time, γ, measurements are performed between agents regularly. The sizes of measurement packets are set to be minimal; therefore, we obtain a baseline measured reference result which is

$$\gamma = \sum_{i=\{EI,IE\}} \{t(i) + p_p(i) + p_c(i) + q(i)\}\,. \tag{1}$$

Hence for a transaction with content information, the size of a packet may vary with additive transmission and processing delays. The total delay is then

$$T = t(RE) + t(ER) + p_p(RE) + p_p(ER) + \gamma + \sum_{i=\{EI,IE|data\}} \{t(i) + p_c(i)\}\,. \tag{2}$$

The $t(RE)$ is also negligible for the sent request message from client. Since the downlink is the bottleneck, we have $t(ER) = \frac{s_A}{B}$ upon carrying out content adaptation. Hence, we can bound the the total delay given in Eqn. (2), i.e.,

$$W > \frac{s_A}{B} + p_p(RE) + p_p(ER) + \gamma + \sum_{i=\{EI,IE|data\}} \{t(i) + p_c(i)\}\,. \tag{3}$$

Therefore, we obtain the resulting estimated size of content adapted packets which is

$$s_A < B \cdot \{W - p_p(RE) - p_p(ER) - \gamma - \sum_{i=\{EI,IE|data\}} [t(i) + p_c(i)]\} \tag{4}$$

$$< B \cdot \{W - \gamma\}\,. \tag{5}$$

To deliver content in real-time, the total delay across the networks should be smaller than W as shown in Eqn. (3). Acutally, the sizes and number of packets sending through the networks may be reduced noticeably. The extra processing delay can possibly be predictable from a user's profile. Indeed, the ingress agent arranges proper operations within networks based on the calculated upper bound on s_A as shown in Eqn. (4). By reducing content size as indicated, real-time delivery of information can be achievable.

3 Experiments and Discussions

A prototype testbed has been set up to validate the adaptive designs of the pervasive network infrastructure. All routers are Pentium III computers. The platform is implemented using active network socket programming (ANSP) interfaces [7] which basically are a set of Java APIs for the ease of protocol implementations.

3.1 Content Adaptation for Real-Time Delivery

The goal of the experiments is to examine the performance of real-time adaptation using the pervasive network infrastructure. When a mobile user moves, the access interface (the bottleneck link) of the device to the Internet changes. Indeed, the network infrastructure is the best medium to detect and monitor these changes in user access context. In order to carry out experiments flexibly, a `tc` script on class-based queueing (CBQ) in Linux is used to emulate the effect of varying last mile bandwidth.

Real-Time Delivery of Web Pages. In order to facilitate real-time delivery, selected information content of web pages can be pre-fetched and compressed. In the experiments, deliveries of web pages are tested against different user's expected real-time contraint W. The constraint measures from the instant that a user send a request for a web page till the instant that the page is displayed.

Recalling that the size of packet adapted to a bottleneck link, s_A, in Eqn. (4), then a loose upper bound in Eqn. (5) can be used as $\sum_{i=\{EI,IE|data\}} [t(i) + p_c(i)]$ cannot be estimated. But, if these transmission and processing delays can be measured, then a tighter bound should be deployed.

In the testbed, 100 Mbps switched Ethernet connections are used. If a packet is delivered along path i and its size is s_i bytes, then $t(s_i) = \frac{8 \cdot s_i}{100 \times 10^6} = \frac{s_i}{12{,}500{,}000}$. Even though the routers are using store-and-forward and there are m_i routers in a path, the transmission delay of $m_i \cdot t(s_i)$ is not significant as the value of m_i is usually small. On the other hand, the compression operations on content may be a time-consuming process in the experiments. Actually, it has been measured in the testbed, and it is equal to 1 msec per 1340 bytes of original bitmap data. Therefore, we have $p_c(s_i) = \frac{s_i}{1{,}340{,}000}$. Furthermore, the sum of $p_p(RE)$ and $p_p(ER)$ is found to be negligible. Therefore, the real-time constraint in the web page delivery experiments becomes

$$s_A(s_i, m_i) < B \cdot \{W - \gamma_i - m_i \cdot t(s_i) - p_c(s_i)\}. \tag{6}$$

Suppose that there are k slices and the set of selected paths is K, $|K| = k$. The path with the longest delay should have impact on the performance of real-time delivery issue. Hence, we have

$$s_A < B \cdot \left\{ W - \max_{i \in K} \{\gamma_i - m_i \cdot t(s_i) - p_c(s_i)\} \right\}. \tag{7}$$

But if the routers along do not carry out the extra processing function, e.g., compression, then $p_c(s_i) = 0$.

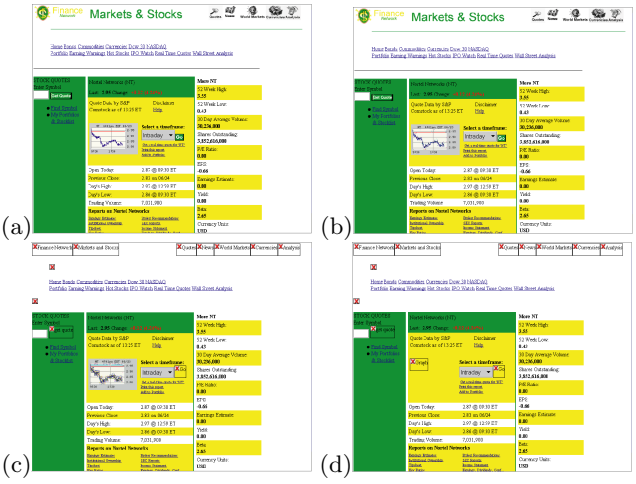


Fig. 3. Real-time delivery (a) original page, (b) moderately compressed (the stock graph), (c) heavily compressed (removals of R_3 components), (d) most components dropped but critical data (the stock price)

A stock-quote web page is used in the experiments, which contains a number of intra-content components and they are ranked into three classes only.

- The text, made up of HTML markup tags and the body of information, contains information essential to a user including current share price, daily high/low, trading volume, and so on. The text is classified as rank 1 (most important) and is always included in deliveries;
- A stock price graph is classified as rank 2. It can be compressed to 397 different sizes when needed. The relationship between compressed size and compression parameter is encoded as the meta-information of the bitmap file. Then, APNI can choose the closest adapted size, and the nodes operate appropriate compression accordingly. Compressed sizes at lower compression parameters have a larger granularity (i.e. more discrete in compressed sizes among higher compression levels). After the maximum compression, the graph has a size of 13590 bytes (37.79% of the original). If the s_A left after rank 1 component is less than the maximally-compressed size, the stock price graph is dropped.
- Graphic buttons and banners are classified as rank 3.

By calculating the desired adapted size s_A with Eqn. (7), a selection of in-page components is obtained. Ideally, the resultant size should vary linearly with s_A. However, this is not possible since the adaptation granularity is non-continuous. Conversely, the resultant adapted size should stay below the ideal curve such that the real-time delivery can be achieved with the maximum amount (i.e. perceptual value) of information delivered to the user. Please note that when s_A is below6600, both the ideal and real curves stay flat. This is because

rank 1 components are always transmitted to preserve the minimum amount of perceptual value for every page.

A stock-quote web page is used in the experiments, which contains a number of intra-content components and they are ranked into three classes, $R_1 > R_2 > R_3$. The sizes of packets are adapted according to the Eqn. (4) as both the $t(\cdot)$ and $p_c(\cdot)$ are measured and calculated.

Fig. 3(a) shows the original web page. Fig. 3(b) shows a resulting page that the R_2 components are compressed for preserving R_3 components. When the bottleneck bandwidth or the real-time constraint is further reduced, the R_3 components are dropped and the R_2 components may further be compressed in Fig. 3(c). In Fig. 3(d), extreme condition occurs and no further image reduction can be possible. Thus, only the R_1 component, i.e., the text file, is delivered.

4 Conclusions

In the paper, we explore the possibility of offering real-time content adaptation on set of data streams using the active pervasive network infrastructure. With different relative importance among data sets, traffic control and discrimination with different operations on content adaptations have been examined. Piggyback extension to users' preferences messages is proposed to smoothly enhance the pervasive network infrastructure design. Content adaptation is achieved transparently to both clients and server systems. Real-time delivery services overcome stochastic network situations and abruptly changing bottleneck link bandwidth problem while retaining information integrity and preserving critical data at the best of the limit of an environment.

References

1. Weiser, M.: The computer for the twenty-first century. Scientific American **265:3** (1991) 94–104
2. Satyanarayanan, M.: Pervasive computing: vision and challenges. IEEE Personal Communications **8:4** (2001) 10–17
3. Banavar, G., Bernstein, A.: Software infrastructure and design challenges for ubiquitous computing. Communications of the ACM **45:12** (2002) 92–96
4. Law, K.L.E., So, S.: Pervasive computing on active networks. The Computer Journal **47:4** (2004) 418–431
5. Chan, A.L., Law, K.L.E.: QoS Negotiations and real-time renegotiations for multimedia communications. IEEE Inter. Conf. Computer Communications and Network 2002, Miami, USA (2002) 522–525
6. Tennenhouse, D., Smith, J.M., Sicoskie, W.D., Wetherall, D.J., Minden, G.J.: A Survey of Active Network Research. IEEE Communications Mag. (1997) 80–86
7. Law, K.L.E., Leung, R.: A Design and implementation of active network socket programming. Microprocessors and Microsystems Journal, Elsevier Publisher **27** (2003) 277–284
8. Postel, J. (ed.): Transmission Control Protocol. IETF, RFC 793, Sept. 1981.

A Semantic Web-Based Infrastructure Supporting Context-Aware Applications

Renato F. Bulcão-Neto[1], Cesar A.C. Teixeira[2], and Maria da Graça C. Pimentel[1]

[1] Universidade de São Paulo, São Carlos-SP, 13560-970, Brazil
[2] Universidade Federal de São Carlos, São Carlos-SP, 13565-905, Brazil
{rbulcao, mgp}@icmc.usp.br, cesar@dc.ufscar.br

Abstract. There is a demand for efforts that deal with the challenges associated to the complex and time-consuming task of developing context-aware applications. These challenges include context modeling, reuse and reasoning, and software infrastructures intended to context management. This paper presents a service infrastructure for the management of semantic context called Semantic Context Kernel. The novelty is a set of semantic services that can be personalized according to context-aware applications' requirements so as to support the prototyping of such applications. The Semantic Context Kernel has been built upon an ontological context model, which provides Semantic Web abstractions to foster context reuse and reasoning.

1 Introduction

One of the research themes in ubiquitous computing[1] is the context-aware computing, where applications customize their behavior based on context information sensed from those instrumented environments. A classic definition of context [1] is "any relevant information about the user-application interaction, including the user and the application themselves". For instance, by means of sensors network and computers, the PROACT elder care system infers whether and how people with early-stage cognitive decline perform activities of daily living [2].

There is a demand for efforts that deal with the challenges associated to the complex and time-consuming task of developing context-aware applications [3]. Challenges related to the development of such applications include: (i) how to represent context in such manner that facilitates its sharing, reuse and processing; (ii) the development of software infrastructures to support applications in respect with context management. High-level context models need representation languages that use broadly accepted standards so as to facilitate the sharing and the reuse of context [4]. Moreover, the more formal a context model is, the better is the ability for context-aware applications to reason about context. A software infrastructure built on top of such context models can then provide context-aware applications with enhanced-services intended to exploit context sharing, reuse and reasoning.

[1] The term "ubiquitous computing" is hereafter referred to "ubicomp".

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 900–909, 2005.
© IFIP International Federation for Information Processing 2005

The literature has reported that the Semantic Web vision [5] fits well the need for context models that enables applications to process the semantics of context even regardless applications domains. The GaiaOS middleware is able to reason about context represented as first-order predicates [6]. The CoBrA agent architecture acquires, manages and reasons about shared ontological context, and also detects and resolves inconsistent context [4]. The Semantic Spaces infrastructure supports the inference of higher-level contexts from basic contexts in which the semantics of context is also explicitly conveyed by ontologies [7]. Although those efforts address important issues such as context classification and dependency, and quality and privacy of context, none have focused on providing applications with semantic services that can be configured according to applications' requirements.

This paper presents a Semantic Web-based service infrastructure for context management called *Semantic Context Kernel*. Its architecture is composed of configurable semantic services for context storage, query and reasoning, and service discovery. The Semantic Context Kernel has been built upon an ontological context model [8] that provides a general vocabulary so lower ontologies can import it for particular domains. The design space for building this context model derives from dimensions for context modeling debated in the literature: identity (*who*), location (*where*), time (*when*), activity (*what*) [9] and devices (*how*) [10]. In order to address these five context dimensions, concepts of well-known semantic web ontologies have been reused and extended.

The novelty regarding the Semantic Context Kernel relies on its configurability feature: services can be personalized according to context-aware applications' requirements so as to facilitate the prototyping of such applications. For instance, the context persistence service allows application developers to choose the type of persistent storage (e.g. files and databases) and the type of serialization [11] (e.g. RDF/XML and N-Triples) for semantic context. Different levels of inference over context can be exploited by means of the context inference service (e.g. transitive and rules-based reasoning).

Section 2 outlines Semantic Web standards that we have exploited to build our context model. In Section 3 we present our ontology-based context model. Section 4 describes the architecture of the Semantic Context Kernel. Section 5 illustrates the use of that infrastructure by an application on the educational domain. Finally, in Section 6 we present concluding remarks and future work.

2 Semantic Web Background

In order to exploit the full potential of the Semantic Web, there is a need for standards for describing resources — anything that can be uniquely identified — in a language that makes their meaning explicit. In order to explicitly associate meaning for data, knowledge must be represented in some way. One attempt to apply ideas from knowledge representation is the RDF standard [12].

The RDF specification provides a generic data model that consists of nodes connected by labeled arcs: nodes represent resources, and arcs represent properties or relations used to describe resources. A resource together with a property and the value of that property for that resource is called an RDF statement. Those three individual parts of a statement form the RDF triple model. The RDF Schema language [13] con-

veys the semantics of RDF metadata by means of mechanisms for describing classes of resources, relationships between resources, and restrictions on properties.

The OWL language [14] is a step further for the creation of controlled, shareable, and extensible vocabularies. OWL builds on RDF and RDF Schema and adds more vocabulary for describing properties and classes, including among others: relations between classes (e.g. disjointness, inverse), (in)equality between individuals, cardinality (e.g. exactly one, at least one), richer typing of properties (e.g. enumerated datatypes), characteristics of properties (e.g. symmetry, transitivity, functional), and constraints on properties (e.g. all values from, some values from, cardinality).

The next section describes our domain-independent ontological context model. It is based on the OWL ontology language due to the following reasons: (i) it provides formal semantics for reasoning about context; (ii) it is compatible with several standard specifications intended to the Semantic Web; (iii) it allows ontologies to be distributed across many systems; and (iv) its openness and extensibility.

3 An Ontological Context Model for UbiComp Applications

Figure 1 depicts our ontological context model described elsewhere [8]. It represents the basic concepts of actors, location, time, activities, and devices as well as the relations between these concepts. Concepts of semantic web vocabularies have been borrowed in order to address every dimension as well as to serve as guidance for context modeling. We describe our context model as follows.

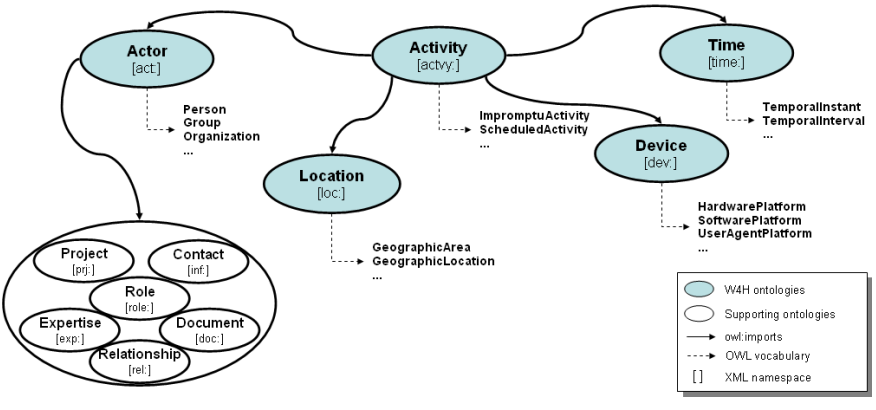


Fig. 1. A domain-independent semantic context model with high-level ontologies

In **Figure** 1, the *Actor* ontology models the profile of entities performing actions in a ubiquitous computing environment such as people, groups and organizations. This ontology imports other ontologies that we have built to deal with actors' profile: *knowledge*, *social relationship*, *document*, *social role*, *contact information* and *project*. The knowledge ontology models knowledge areas so as to relate someone to a particular

expertise or topic of interest. The relationship ontology describes people's social network (e.g. *cooperatesWith*). The document ontology models documents made by actors such as web pages. The role ontology describes the actors' social role in the real world (e.g. *student*). The contact ontology represents different types of actors' contact information (e.g. *email*). Finally, the project ontology models meta-information associated to projects and actors (e.g. *isHeadedBy*).

The *Location* ontology describes the whereabouts of actors. It models indoor and outdoor places (e.g. *room* and *parking lot*), containment and spatial relations between places (e.g. *isPartOf* and *isConnectedTo*), and geographic coordinates (e.g. *latitude* and *longitude*). This ontology also represents places with respect to the address that they are located (e.g. *zip code*). In other words, a building can be related to the address where it is located (e.g. *street* and *zip code*).

The *Time* ontology represents time in terms of temporal instants and intervals [15]. We modeled temporal relations between instants and intervals (e.g. an instant is *insideOf* an interval), properties of intervals (e.g. the *durationOf*), and temporal relations between intervals (e.g. *equals*, *starts* and *finishes*). The temporal ontology also provides a standard way of representing calendar and clock information on the Semantic Web.

The *Device* ontology describes devices features regarding its *hardware*, *software* and *user agent* platforms. The hardware platform describes the I/O and network features of a device (e.g. whether a display is color-capable). The software platform describes application environment, operating system, and installed software (e.g. the types of Java virtual machines supported). The user agent platform describes the web browser running on a device (e.g. whether it is Javascript-enabled).

The *Activity* ontology describes actions that actors do or cause to happen. We modeled an activity as a set of relevant events that characterizes it. An event is a fact that includes *spatiotemporal* descriptions, as well as descriptions of the corresponding *actors* and *devices* involved. The relevance, the type and the combination of events for inferring activities are dependent on the user's task in the current domain.

The next section presents the service infrastructure that we have built for the management of semantic context.

4 The Semantic Context Kernel

Built upon our semantic context model, we have implemented a service infrastructure called *Semantic Context Kernel*. The aim is to provide developers with a set of semantic-enabled services that can be configured so as to address applications' requirements. Since this work handles the semantics of context, it furthers our previous work on software infrastructure for context awareness [16]. **Figure** 2 depicts the Semantic Context Kernel architecture.

In **Figure** 2, context information is provided by *context sources*, which include applications, web services, and physical sensors. *Context transducers* convert the information captured from context sources into a common semantic representation: the RDF triple model. This approach addresses both interoperability and reuse issues. *Context consumers* make use of context information stored by context sources so that the former can adapt themselves following the current situation (e.g. applications). The *dis-*

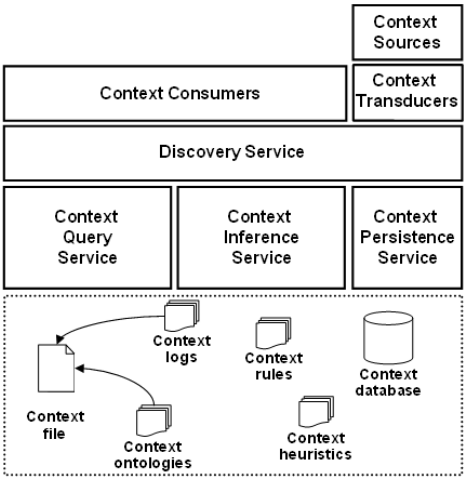


Fig. 2. The Semantic Context Kernel architecture

covery service provides context transducers and every service layer with an advertising mechanism so as to allow context consumers to locate these services.

The *context query service* allows context consumers to query context through a declarative language for RDF models that support simple conjunctive triple patterns called RDQL (*RDF Data Query Language*) [17]. In the general case, query expressions are represented as a matching of a triple pattern against an input source RDF graph. Example 1 describes an RDF query declared by a context consumer so as to obtain all sequence of triples matching the following constraint: the list of names (variable *?name*) and corresponding chat IDs of type ICQ (variable *?icqValue*) of all people with some resource whose name is "Steve Orr" works. The result of the current query is as follows: "Ian Battle" and "10043355".

```
<!--            Example 1            -->
SELECT ?name, ?icqValue
FROM   <file:sck/contextFile.nt>
WHERE  (?x <act:hasName> "Steve Orr")
       (?x <rel:worksWith> ?y)
       (?y <act:hasName> ?name)
       (?y <act:hasContactProfile> ?z)
       (?z <inf:imType> "ICQ")
       (?z <inf:imValue> ?icqValue)
USING  act for <http://linkserver/2005/actor.owl#>
       rel for <http://linkserver/2005/relationship.owl#>
       inf for <http://linkserver/2005/contactinfo.owl#>
```

The input source is a file (*FROM* clause) containing an RDF graph with all triples (see **Figure** 3) representing context information stored by context sources. In this case, the file *contextFile.nt* stores RDF triples using an alternative serialization called N-Triples, where each RDF triple is represented in a line-based, plain text format. Example 2 is the content of the context repository represented in N-Triples, which is a very suitable serialization for large RDF models.

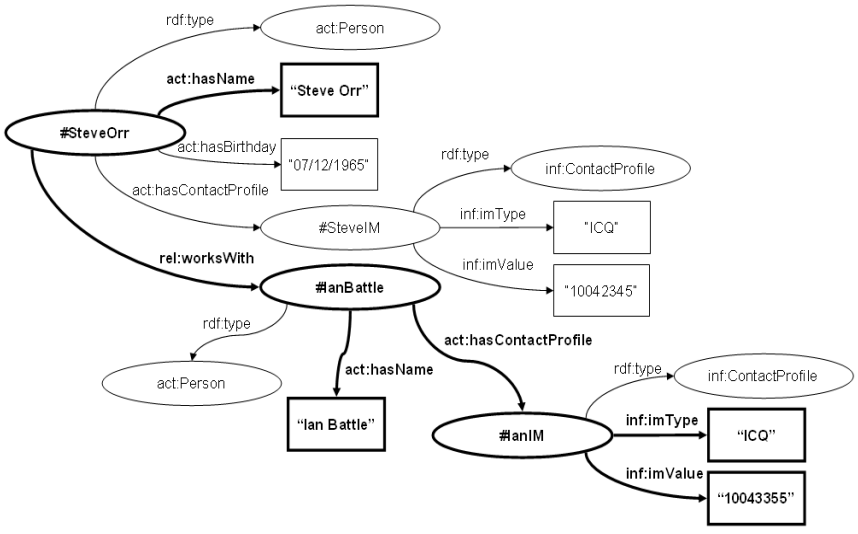


Fig. 3. This RDF graph represents the content of the context file identified by the FROM clause in the RDF query described in Example 1. The triples described in that query are printed in bold.

```
<!--            Example 2            -->
@prefix : <http://linkserver/example#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix act: <http://linkserver/2005/actor.owl#>.
@prefix rel: <http://linkserver/2005/relationship.owl#>.
@prefix inf: <http://linkserver/2005/contactinfo.owl#>.
:SteveOrr rdf:type act:Person.
:SteveOrr act:hasName "Steve Orr".
:SteveOrr act:hasBirthday "07/12/1965".
:SteveOrr act:hasContactProfile :SteveIM.
:SteveOrr rel:worksWith :IanBattle.
:SteveIM  rdf:type inf:ContactProfile.
:SteveIM  inf:imType "ICQ".
:SteveIM  inf:imValue "10042345".
:IanBattle rdf:type act:Person.
:IanBattle act:hasName "Ian Battle".
:IanBattle act:hasContactProfile :IanIM.
:IanIM  rdf:type inf:ContactProfile.
:IanIM  inf:imType "ICQ".
:IanIM  inf:imValue "10043355".
```

The *context inference service* provides context consumers with a configurable inference support over context. Developers can specify the level of inference to be supported over context, e.g. a transitive reasoner basically infers the hierarchical relations between classes (via *rdfs:subClassOf*). Example 3 shows some results of an inference process using a transitive reasoner over the RDF graph depicted in **Figure** 4.

```
<!--            Example 3            -->
Resource SteveOrr is instance of the class act:Person;
                                           act:Actor;
Resource SteveLocation is instance of the class loc:MeetingRoom;
                                                loc:Room;
                                                loc:IndoorLocation;
```

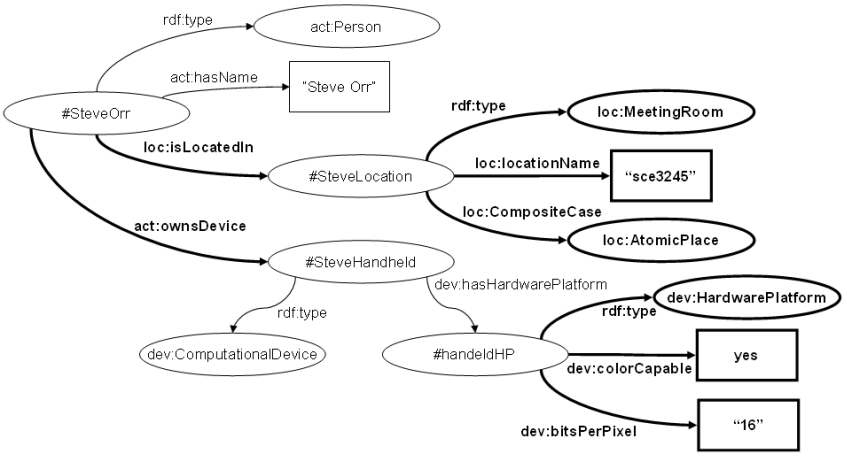


Fig. 4. This RDF graph describes a person called Steve Orr, located in a meeting room called sce3245, with a handheld device with color capability. Context information printed in bold depicts the relation between actors, locations and devices, and the characteristics of locations and devices.

On the other hand, in order to exploit high-level context, developers can define rules and store them on files, called *context rules*. When the context inference service is set up to use rules, it reads ontology facts into memory represented as RDF triples and parse those rules so as to validate them. When using rules, the context inference service allows developers to choose the type of reasoning to be performed, e.g. inductive, deductive and inductive-deductive reasoning. Example 4 presents two forward-chaining rules: the former describes that if a graduate student and his supervisor are in the meeting room, then they are attending a meeting; the latter describes that if two graduate students are members of a same study group and both are in the study room using the same tablet PC, then they are attending a study group meeting.

```
<!--            Example 4            -->
Person(A) ^ hasRole(A,B) ^ Person(C) ^ hasRole(C,D) ^ Graduate(B) ^ Faculty(D) ^
isSupervisorOf(D,B) ^ isLocatedIn(A,E) ^ isLocatedIn(C,E) ^ MeetingRoom(E)
--> attendingMeeting(A,C)

Person(A) ^ hasRole(A,B) ^ Person(C) ^ hasRole(C,D) ^ Graduate(B) ^ Graduate(D) ^
isLocatedIn(A,E) ^ isLocatedIn(C,E) ^ StudyRoom(E) ^ isMemberOf(A,F) ^
isMemberOf(C,F) ^ StudyGroup(F) ^ ownsDevice(A,G) ^ ownsDevice(C,G) ^ TabletPC(G)
--> StudyGroupMeeting(A,C)
```

If the inference process over context originates a conflict, user-defined *context heuristics* can be used to resolve it. An example of conflict can arise when a person is located in two different atomic places at the same time, e.g. a meeting room and a classroom with no relation of composition between them. This can be originated due to problems with freshness and accuracy of location information gathered from physical sensors.

The *context persistence service* allows developers to choose the types of persistent storage and context serialization. We allow to store context in relational databases as well as on a *context file*. The former approach can handle context in the RDF/XML and the N-Triples syntaxes, whereas the latter represents context in N-Triples.

The file-based approach is an alternative for applications that do not require functionalities of databases such as consistency of data or transactions. Context log files store every new RDF model collected from context sources on a regular basis (also configured by developers). Afterwards, these files are merged into the persistent context file. Both the triples representation of context and the content of ontologies are stored on the context file. Otherwise, when storing context on databases, ontologies are stored on separate files and read when necessary only (e.g. by the context inference service).

5 Evaluation with an Application on the Educational Domain

TIDIA-Ae is a project aiming at developing and deploying an e-learning infrastructure that exploits a large area high speed Internet network. The basic conceptual model of the project is a core *State* class which may contain recursively other *State* classes and: (a) the *State* class has associations with the *User* class; (b) a *User* may have varied *Roles* (instances of the *Role* class) in different *States*; (c) *Tools* are made available to users when in a given *State*; (d) the *State* class has relationships with the *Contents* class — the aim is to allow *Users* to access different *Contents* and *Tools* in a given *State* depending on their *Role*.

As designed, a typical use of the TIDIA-Ae infrastructure is as follows: a *User* authenticates to enter in a *State*; in that *State* the *User* has a pre-defined *Role* (say *Instructor*) which gives access to pre-defined *Contents* and *Tools*. One of the available tools may be a *Collaborative Editor* which allows new *Contents* to be generated. Other available tools may be an *Instant Messenger*, which allows several participants to communicate synchronously, or a *Whiteboard* tool which, running on a large electronic whiteboard or on portable tablet PCs, may be used to deliver a class either in a traditional classroom or laboratory setting (with students and instructor in the same room) or a distributed mode (with participants in different physical locations). Moreover, the information captured by the *Whiteboard* tool can be used to generate new *Contents* for that *State*.

Considering that the Semantic Context Kernel has been designed to support building context-aware applications, we have investigated its use to allow context-dependent operations to be integrated with the TIDIA-Ae infrastructure. Such integration would allow rules and queries supporting services such as:

- An instructor (*actor* with a *Role*) wishes to be notified (via a validated *rule*) when a given number of students (*actor*s with other *Role*s) is engaged in a conversation supported by the *Instant Messenger* (*software platform*) in any of the courses (*Activity*) he is responsible for. Moreover, this may be associated with an inference that, if the students are also viewing the same *Contents* (e.g. from the *document ontology*), they are in a *Study Group Meeting*.
- A participant (instructor or student, an *actor* with a *Role*) wishes to be notified when some new *Contents* is created via the *Collaborative Editor* (*software platform*) or via the use of the *Whiteboard* tool (*software platform*) in any of the courses (*Activity*) he is involved with — the rule may also specify that the notification may occur only when a tablet PC is used as *hardwarePlaftorm*.

From our investigation so far, we have already learned that most of the information we need for this application can be modelled by the following ontologies: actors, time, activities, and devices. However, we have also identified that the location ontology should be extended to support both physical and virtual locations: if a *Location* could be a virtual location, a *State* would be modelled as a virtual location. We have also identified that some of the ontologies we deal with an *Actor* profile, in particular *Document* and *Project*, may be alternatively associated with an *Activity* as well.

Finally, both the context query service and the context inference service would be used to support high-level services.

6 Concluding Remarks

Ubiquitous computing applications must have access to context information in order to be able to adapt their services accordingly to users' needs. We illustrated our vision by means of the Semantic Context Kernel, an ongoing project that provides semantic-enhanced services for the management of context information gathered from ubicomp environments. The main contribution includes semantic services that can be customized following context-aware applications' requirements toward making it easier the task of development of such applications. These semantic services allow applications not only to store and query context, but also to reason about context in a configurable fashion.

The Semantic Context Kernel has been built on top of an ontological context model for ubicomp applications. This context model has borrowed concepts from consensus Semantic Web ontologies due to the amount of information that they describe. Importing of such ontologies would overload the process of inference over context, even though being an in-memory process.

We have also illustrated the use of the Semantic Context Kernel from the perspective of an application on the educational domain, which demonstrates its value at the same time that allows the identification of possible extensions.

Some key points about context have not been considered yet in our work such as inconsistence, freshness, and privacy of context. We assume context sources as accurate and updated context providers. Privacy of context is also a serious issue on context-aware computing [18]. For instance, we should support privacy of location information when context consumers need to track someone's whereabouts.

Regarding the Semantic Context Kernel infrastructure, future work includes a plug-in support to external inference engines so as to increase the configurability of the context inference service. The context query service can also be extended to support SPARQL queries [19], a W3C effort for a standard query language for RDF.

Acknowledgments

Renato Bulcão Neto is PhD candidate supported by FAPEMA (03/345). Maria Pimentel and Cesar Teixeira are supported by FAPESP and CNPq. This work is supported by Hewlett-Packard, in the context of the *Applied Mobile Technology Solutions in Learning Environments Project*, and is funded by FAPESP in the context of the TIDIA-Ae Project (http://tidia-ae.incubadora.fapesp.br).

References

1. Dey, A.K.: Understanding and using context. Personal and Ubiquitous Computing Journal **5** (2001) 4–7
2. Philipose, M., Fishkin, K.P., Perkowitz, M., Patterson, D.J., Fox, D., Kautz, H., Hahnel, D.: Inferring activities from interactions with objects. IEEE Pervasive Computing **3** (2004) 50–57
3. Helal, S.: Programming pervasive spaces. IEEE Pervasive Computing **4** (2005) 84–87
4. Chen, H., Finin, T., Joshi, A.: Semantic Web in the context broker architecture. In: Proceedings of the International Conference on Pervasive Computing and Communications. (2004) 277–286
5. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American **284** (2001) 35–43
6. Ranganathan, A., Campbell, R.H.: An infrastructure for context-awareness based on first order logic. Personal Ubiquitous Computing **7** (2003) 353–364
7. Gu, T., Pung, H.K., Zhang, D.Q.: A service-oriented middleware for building context-aware services. Journal of Network and Computer Applications **28** (2005) 1–18
8. Bulcão Neto, R.F., Pimentel, M.G.C.: Toward a domain-independent semantic model for context-aware computing. In: Proceedings of the 3rd Latin American Web Congress, IEEE Press (2005) 61–70
9. Abowd, G.D., Mynatt, E.D., Rodden, T.: The human experience. IEEE Pervasive Computing **1** (2002) 48–57
10. Truong, K.N., Abowd, G.D., Brotherton, J.A.: Who, what, when, where, how: Design issues of capture & access applications. In: Proceedings of the International Conference on Ubiquitous Computing. (2001) 209–224
11. Beckett, D.: RDF/XML syntax specification (revised) (2004) http://www.w3.org/TR/rdf-syntax-grammar/.
12. Klyne, G., Carroll, J.J.: Resource Description Framework (RDF): concepts and abstract syntax (2004) http://www.w3.org/TR/rdf-concepts/.
13. Brickley, D., Guha, R.V.: RDF vocabulary description language 1.0: RDF Schema (2004) http://www.w3.org/TR/rdf-schema/.
14. Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: OWL Web ontology language reference (2004) http://www.w3.org/TR/owl-ref/.
15. Bulcão Neto, R.F., Pimentel, M.G.C.: Semantic interoperability between context-aware applications. In: Proceedings of the Brazilian Symposium on Multimedia and Web Systems. (2003) 371–385 (In Portuguese).
16. Bulcão Neto, R.F., Jardim, C.O., Camacho-Guerrero, J.A., Pimentel, M.G.C.: A web service approach for providing context information to CSCW applications. In: Proceedings of the 2nd Latin American Web Congress, IEEE Press (2004) 46–53
17. Miller, L., Seaborne, A., Reggiori, A.: Three implementations of SquishQL: a simple RDF query language. In: Proceedings of the International Semantic Web Conference. (2002) 423–435
18. Lahlou, S., Langheinrich, M., Rocker, C.: Privacy and trust issues with invisible computers. Communications of the ACM, Special issue: The disappearing computer **48** (2005) 59–60
19. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF (2005) http://www.w3.org/TR/rdf-sparql-query/.

A Universal PCA for Image Compression

Chuanfeng Lv and Qiangfu Zhao

The University of Aizu. Aizuwakamatsu, Japan 965-8580
{d8061105, qf-zhao}@u-aizu.ac.jp

Abstract. In recent years, principal component analysis (PCA) has attracted great attention in image compression. However, since the compressed image data include both the transformation matrix (the eigenvectors) and the transformed coefficients, PCA cannot produce the performance like DCT (Discrete Cosine Transform) in respect of compression ratio. In using DCT, we need only to preserve the coefficients after transformation, because the transformation matrix is universal in the sense that it can be used to compress all images. In this paper we consider to build a universal PCA by proposing a hybrid method called k-PCA. The basic idea is to construct k sets of eigenvectors for different image blocks with distinct characteristics using some training data. The k sets of eigenvectors are then used to compress all images. Vector quantization (VQ) is adopted here to split the training data space. Experimental results show that the proposed approach, although simple, is very efficient.

1 Introduction

So far many techniques have been proposed for image compression. These techniques can be roughly divided into two categories: predictive approaches and transformational ones. In brief, predictive approaches like differential pulse code modulation (DPCM) and vector quantization (VQ) try to predict a pixel or a block of pixels based on known data (already observed or previously stored). Usually, only local prediction is considered. For example, in DPCM, good prediction can be made even if the predictor is very simple because neighboring pixels are often highly correlated. In VQ, a block of pixels can be predicted very well using the nearest code word.

Transformational approaches project the data into a domain which requires fewer parameters for data representation. Principal component analysis (PCA) is known as the optimal linear transformation for this purpose. Compared with VQ which approximates each point in the problem space using a different code word, PCA approximates all points using the linear combinations of the same set of basis vectors. Thus, we may consider VQ and PCA as two extreme cases. VQ is an extremely local approach which approximates each point using only one point (the nearest code word), while PCA is an extremely global approach which approximates all points using the same set of basis vectors. So far PCA has been successfully adopted in signal processing, image processing, system control theory, communication, pattern recognition, and so on. PCA can be used to compress the dimensionality of the problem space. PCA achieves compression through discarding the principle components with small eigenvalues. However, since the compressed data must include both the

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 910 – 919, 2005.
© IFIP International Federation for Information Processing 2005

transformation matrix (the eigenvectors) and the transformed coefficients, PCA cannot produce high compression ratio.

Another transformation for image compression is DCT (Discrete Cosine Transform). Although DCT is not optimal, it is one of the most popular transforms, and has been used and studied extensively. The important feature of DCT is that it takes correlated input data and concentrates its energy in just the first few transformed coefficients. The advantage of using DCT is that we need only to preserve the transformed coefficients, since the transformation matrix is universal in the sense that it can be used to compress all images. Clearly, a PCA encoder build from one image cannot be used to compress all other images because the eigenvectors obtained from one image cannot approximate other images well. Actually, even if we consider the same image, the PCA encoder usually cannot approximate all image blocks equally well using a fixed set of eigenvector vectors. It may perform poorly in local regions which include edges or noises.

To increase the approximation ability, many improved PCA approaches have been proposed in the literature [7], [8]. The basic idea of these approaches is to train a number of PCAs which can adapt different image blocks with distinct characteristics. Though these algorithms can improve conventional PCA in some extend, but they are very time consuming and cannot be used easily.

In this paper, we propose a new approach named k-PCA by combining VQ and PCA. The basic idea is to divide the problem space roughly using VQ, and then find a different set of eigenvectors using PCA for each cluster (not sub-space). The point is, if the training data are complicated enough, we can construct a set of universal eigenvectors which can be used to compress any input image. Experimental results show that the proposed k-PCA approach, although simple, outperforms existing methods in the sense that the reconstructed images can have better quality without decreasing the compression ratio.

This paper is organized as follows: Section 2 provides a short review of VQ and PCA, and introduces briefly the concept of MPC (mixture of principle component). In Section 3, we propose the k-PCA approach by combining VQ and PCA. The proposed method is verified through experiments in Section 4. Section 5 is the conclusion.

2 Preliminaries

2.1 Vector Quantization (VQ)

VQ extends scalar quantization to higher dimensions. This extension opens up a wide range of possibilities and techniques not present in the scalar case. To implement VQ, the first step is to initialize a codebook based on the input data. The LBG algorithm as a standard approach has been widely adopted in many data compression system [1]. Its main steps are as follows:

Step 0: Select a threshold value a (>0), set k=1, and set the mean of all input data (the training data) as the first code word: $C_k^{(1)}$ (were k=1).

Step 1: If k is smaller than the pre-specified codebook size, continue; otherwise, terminate.

Step 2: Split each of the current code words into two by duplicating it with a small noise.
Step 3: Based on the current codebook, calculate the distortion, say e0. For each code word, find all the input data which satisfy:

$$d(B_m, C_i) = \min_j d(B_m, C_j) \tag{1}$$

where B_m $(m \in [1, P])$ is the m-th input datum, and P is the number of input data.
Step 4: Re-calculate each code word as the mean of the input data found in the last step. Based on the new code word, calculate the reconstructed distortion say e_1. If e_0-e_1<a then go to step 1; else go to step 3.

The distortion is often defined as the mean squared error (MSE) given by

$$MSE = \frac{1}{P}\sum_{m=1}^{P} \left\| B_m - C_m \right\|^2 \tag{2}$$

where C_m is the nearest code word for the m-th block B_m. and $\| \ \|$ is the Euclidean distance between two vectors. The distortion can also be defined as the peak signal to noise ratio (PSNR) as follows:

$$PSNR = 10\log_{10} \frac{f_{\max}^2}{MSE} \ \text{(dB)} \tag{3}$$

where $f_{\max}$ is the maximum value of the image. For a gray image with eight bits per pixel, $f_{\max}$ is 255.

After building the codebook, the coding procedure is very simple. For each input datum, find the nearest code word in the codebook. The index of the code word will be the code of this datum. For decoding, iteratively read in the index stream first, substitute each index with the code word, and put it to the image in order.

VQ is a piece-wise-linear approach. It approximates each point locally. It is locally linear but globally non-linear. It uses only one code word for each input vector. In addition, VQ is a pure discrete representation of the data, and thus can achieve high compression ratio. There are mainly two problems in using VQ. The first one is the so called trade off relation between the compression ratio (Cr) and the fidelity. For example, in order to improve Cr, the codebook size needs to be reduced but the fidelity will be decreased. To resolve this problem, we have proposed an iterated function system (IFS) based algorithm in [2], [3]. The second shortage of VQ is the computational cost for building the codebook, which has become a bottleneck for applying VQ. This is actually one of the major research topics for improving VQ. The k-PCA approach proposed in this paper might be one of the promising methods for solving this problem.

2.2 Principal Components Analysis (PCA)

PCA, also known as Karhunen-Loève transformation in communication theory, can maximize the decreasing rate of the variance of the input data, through resolving the eigenvalue problem:

$$Rq = \lambda q \tag{4}$$

where R is the correlation matrix of the input data, λ is the eigenvalue of R, and q is the eigenvector. If the problem space is N-dimensional, we can have N possible solutions for the vector q. The principal components can be defined as follows:

$$a_j =< x, q_j >, \ \ j = 1,2,...,N \tag{5}$$

where a_j denotes the projections of x onto the j-th principal direction. To reconstruct the original data, we simply have

$$x = \sum_{j=1}^{N} a_j q_j \tag{6}$$

Usually, some of the eigenvalues are very small, and the corresponding eigenvectors can be omitted in Eq. (6). This is the basic idea for data compression based on PCA. The more eigenvectors we omit, the higher the compression ratio will be.

2.3 Mixture of Principle Components (MPC)

By implementing PCA we know that, it is one image vs. one transform method, since for each image we should build one particular transformation matrix consisting of eigenvectors. When reconstructing the image, not only the transformed coefficients but also the transform matrix is required. Furthermore PCA is a linear approach; it cannot approximate all areas of the image equally well. In other words, one PCA cannot simultaneously capture the features of all regions. To resolve the above problems, MPC has been studied [7], [8]. The procedure is as follows: before PCA, divide the problem space into a number of sub-spaces, and then find a set of eigenvectors for each sub-space. If enough training data are given, MPC can construct a system which maintains a good generality. It is interesting to note that an MPC can be used as a universal encoder if the generalization ability is high enough. In this case, we do not have to preserve the MPC parameters in the compressed data. Only the transformed coefficients (the output of the system) for each input image block are needed.

So far researches have been focused on how to divide the problem space efficiently. In [7], Donny proposed an optimally adaptive transform coding method. It is composed of a number of GHA neural networks. Fig. 1 illustrates how the appropriate GHA is selected to learn from the current input vector.

The training algorithm is as follows:

Step 1: Initialize (at random) K transformation matrices $W_1, W_2, \cdots, W_K$, where W_j is the weight matrix of the j-th GHA network.

Step 2: For each training input vector *x*, classify it to the i-th sub-space, if

$$P_i x = \max_{j=1}^{K} P_j x \tag{7}$$

where $P_i = W_i^T W_i$ .Update the weights according to the following rule:

$$W_i^{new} = W_i^{old} + \alpha Z(x, W_i^{old}) \tag{8}$$

Where α is the learning rate and *Z* is a GHA learning rule which converges to the principal components.
Step 3: Iteratively implement the above training procedure until the weights are stable.

In [7], the training parameters are: 1) the number of sub-spaces is 64 and 2) the number of training iterations is 80,000. Note that to use the MPC as a universal encoder; we must train it using many data. The above algorithm clearly is not good enough because it is too time consuming. In paper [8], several methods were proposed to speed up the training process and decrease the distortion. These methods include growth by class insertion, growth by components addition and tree structured network. The essential issue is that the convergent speed of GHA is very slow.

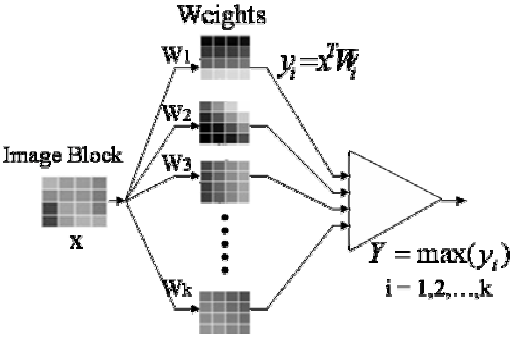


Fig. 1. Basic structure of the MPC

3 The k-PCA

As can be seen from the pervious discussion, the computational cost of the MPC is very high. One reason is that the weight matrices to be updated are of high dimensionality, and another reason is that the convergent speed of the GHAs is slow. To solve these problems, we propose to divide the problem space using VQ. First, the dimension of the vectors (code words) to be updated is much smaller. Second, the LBG algorithm is much faster than the algorithm given in the last section. Third, for

each cluster, we do not use a GHA, but a PCA, and to get a PCA is much faster. The encoding and decoding procedure of the proposed method is given in Fig.2.

Step 1: Divide the input image into $n \times n$ small blocks (n=8 here). For the entire input data, find an 8-D PCA encoder. By so doing we can reduce the dimension of the problem space from 64 to 8.
Step 2: Find a codebook with k (k=64 in our experiments) code words using the LBG algorithm, for the 8-D vectors obtained in the last step, and record the index of each input vector.
Step 3: Based on the codebook, we can divide the problem space into k clusters. For each cluster, we can find an M-D (M=4 in this paper) PCA encoder.
Step 4: For each input vector, compress it to an 8-D vector using the PCA encoder found in Step 1, then find the index of the nearest code word found in Step 2, and finally compress it to an M-D vector. The M-D vector along with the index of the nearest code word is used as the code of the input vector.

The purpose of Step 1 is to reduce the computational cost of VQ. Through experiments we have found that an 8-D PCA encoder can represent the original image very well. The codebook obtained based on the 8-D vectors performs almost the same as that obtained from the original 64-D vectors. In this paper, we call the above encoding method the k-PCA. Note that if we train the k-PCA using enough data, we can use it as a universal encoder, and do not have to include the eigenvectors into the compressed data. Thus, the compression ratio can be increased.

The reconstruction (decoding) procedure is as follows:

Step 1: Read in the codes one by one.
Step 2: Find the basis vectors for the cluster specified by the index, and transform the M-D vector back to the 8-D vector.
Step 3: Transform the 8-D vector back to $n \times n$-D vector, and put it to the image in order.

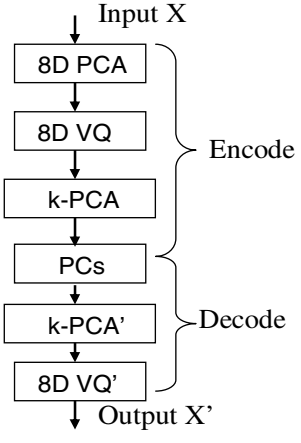


Fig. 2. The flow-chat of the proposed method

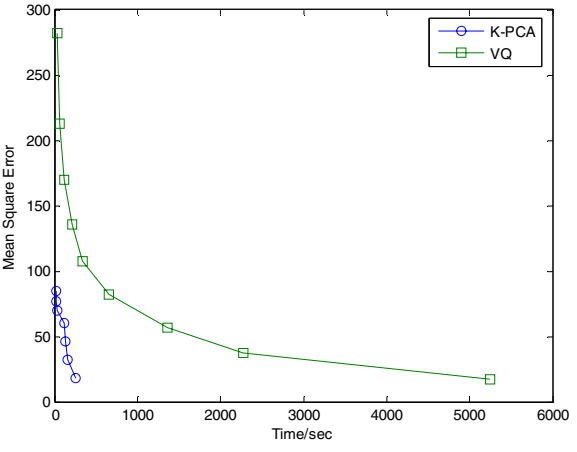


Fig. 3. Training time vs. MSE of VQ and k-PCA. The training image is Lena. Block size is 8*8; VQ is base on LBG algorithm.

4 Experimental Results

To verify the proposed method, we conducted experiments with six popular images. The size of the images is the same, and is 512*512 pixels. There are 256 gray levels. So the uncompressed size of each picture is 256 kB. In the first set of experiments, we constructed the k-PCA using one image and tested the performance using the same image. The experimental results are shown in Table.1. Here, the block size n is 8, the codebook size k is 64, and the number of basis vectors M is 4. Each principal component was quantized to 8bits. The compression ratio in this case is 3.084 (Since the transformation matrix as well as the transformed coefficients are all counted, the compression ratio is quite low). For comparison some experimental results from [8] are also given in table 2, containing only results for the image Lena. In Table 1 and Table 2, the parameters n, k and M are the same. From the last row of Table 1 we can see that the proposed method is better than all results given in Table 2. Note that for MSE, smaller is better. For PSNR, larger is better.

In principle, VQ may achieve higher compression ratio than the proposed method. However, it is usually very time consuming. Fig. 3 shows the relation between the training time and the MSE of VQ and the k-PCA. Obviously, the computational cost for building the codebook is very high, and thus VQ cannot be used in many real-time applications. Thus, in this paper, we use VQ only for finding the k clusters, and the k-PCA is then found based on these clusters.

To confirm the generalization ability of the proposed method, we conducted another set of experiments. Specifically, we used 5 of the 6 images for training, and test the resulted encoder using the remaining image. This method is often called cross-validation in machine learning. The basic idea is that if the training samples are enough, good performance for the test image can be expected. The training and test results PCA and the proposed method are given in Tables 3-4. Two of the reconstructed images which use the proposed method are shown in Fig. 4 and Fig. 5 respectively.

Table 1. Result of the proposed method

	MSE	PSNR
Boat	102.26	27.86
Barbara	151.92	26.32
Lena	46.03	31.50
Mandrill	350.4	22.68
Peppers	55.33	30.71
Zelda	26.42	33.91

Table 2. Results of existing methods

Results for Lena	MSE	PSNR
PCA	75.95	29.3
Growth MPC	57.1	30.06
Tree MPC	57.0	30.05
Standard MPC	84.9	28.8

Table 3. Result of 5D PCA (compression ratio is 12.8)

MSE/PSNR	Training	Test
Boat	185.81/25.44	157.41/26.16
Barbara	162.05/26.03	273.63/23.76
Lena	203.81/25.04	64.91/30.00
Mandrill	120.916/27.31	479.37/21.32
Peppers	197.92/25.17	93.89/28.40
Zelda	210.26/24.90	30.90/33.23

Table 4. Result of proposed method (compression ratio is 13.47)

MSE/PSNR	Training	Test
Boat	56.96/30.57	114.49/27.54
Barbara	55.61/30.67	239.44/24.34
Lena	59.47/30.38	48.15/31.30
Mandrill	22.61/34.58	398.36/22.13
Peppers	91.05/28.53	64.856/30.01
Zelda	59.37/30.39	23.799/34.37

Notice that we are trying to train a set of universal eigenvectors that are good for any image. The BBP (bit per pixel) is only calculated in terms of the transformed coefficients. From the test results, we can see that the proposed method has better generalization ability in all cases, although k-PCA has a little bit higher compression ratio. Of course, we used only five images for training the k-PCA in the experiments. If we use more images, the generalization ability can be further improved. This means that the proposed k-PCA is a very promising method for universal image compression.

Fig. 4. Reconstructed Zelda image in Table 4. The compression ratio is 13.47, the testing MSE and PSNR are 23.799, 34.37. Encoding time is 7.14 Sec, Decoding time is 0.328 Sec.

Fig. 5. Reconstructed Lena image in Table 4. The compression ratio is 13.47, the testing MSE and PSNR are 48.15, 31.30. Encoding time 7.172 Sec, Decoding time is 0.406 Sec.

5 Conclusions

In this paper we have focused our attention on how to improve the performance of PCA based image compress-ion techniques. PCA and several improved PCA methods have been compared through experiments. Based on the respective advantages and

disadvantages, a hybrid approach called k-PCA has been proposed. In this method, a well trained universal eigenvectors act as a common transformation matrix like cosine function in DCT, and the VQ is used to divide the training data into k clusters. A pre-PCA has also been used to reduce the time for building the VQ codebook. We have shown that the proposed method is actually better than all existing methods in respect of reconstruction fidelity, generalization ability and computational cost.

References

1. Y. Linde, A. Buzo and R. M. Gray, "An Algorithm for Vector Quantization," IEEE Trans. On Communications, Vol. 28, No.1, pp.84-95, 1980.
2. C. F. Lv and Q. Zhao, "Fractal Based VQ Image Compression Algorithm," Proc. of the 66th National Convention of IPSJ, Japan, 2004.
3. C. F. Lv, "IFS+VQ: A new method for Image Compression," Master Thesis, the University of Aizu, Japan, 2004.
4. E. Oja, "A simplified neuron model as a principal component analyzer", J. Math. Biology 15, pp. 267-273, 1982.
5. S. Carrato, Neural networks for image compression, Neural Networks: Adv. and Appl. 2 ed., Gelenbe Pub,North-Holland, Amsterdam, 1992, pp. 177-198.
6. T. D. Sanger, "Optimal unsupervised learning in a single-layer linear feedforward neural network", Neural Networks 2, pp. 459-473, 1989.
7. S. Y. Kung and K. I. Diamantaras, "A neural network learning algorithm for adaptive principal component extraction (APEX)", in Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing 90, pp. 861-864, (Al-burqurque, NM), April 3-6 1990.
8. R. D. Dony, "Adaptive Transform Coding of Images Using a Mixture of Principal Components". PhD thesis, McMaster University, Hamilton, Ontario, Canada, July 1995.

An Enhanced Ontology Based Context Model and Fusion Mechanism*

Yingyi Bu, Jun Li, Shaxun Chen, Xianping Tao, and Jian Lv

State Key Laboratory for Novel Software Technology, Nanjing University,
Nanjing City, P.R. China, 210093
{byy, lijun, csx, txp, lj}@ics.nju.edu.cn

Abstract. With diverse sensors, context-aware applications which aim at decreasing people's attentions to computational devices are becoming more and more popular. But it is inadequate just with sensors because what applications really need is high-level context knowledge rather than low-level raw sensor data. So a software layer which delivers high quality contexts to applications with an easy application programming model is needed. In this paper, we establish a formal context model using semantic web languages and design a context fusion mechanism which not only generates high-level contexts by reasoning, but also brings in context lifecycle management and periodically time based conflict resolution to improve the quality of contexts. Using the context fusion mechanism, a programmable and service oriented middleware is built upon OSGi framework to support context-aware applications. Also, we propose an application programming model using RDQL as context query language and demonstrate an application called Seminar Assistant. According to the experiment results, we believe our prototype system is useful for non-real-time applications in various domains.

1 Introduction

Compared with desktop computing, the most remarkable advantage of ubiquitous computing is context-awareness. Context refers to any information that can be used to characterize the situation of entities (i.e. whether a person, place or object) [1]. Computational entities in ubiquitous computing environment need to be context-aware so that they can follow changing situations and provide personalized services according to different situations. In this way, people's attentions to diverse computational devices can be largely decreased so that computers may "weave themselves into the fabric of everyday life until they are indistinguishable from it" [2].

To achieve context-awareness, applications should obtain semantic contexts meeting their needs. However, what sensors provide are just low-level physical data which have no semantic meanings so that applications should convert those sensor data to high-level context knowledge themselves. We think the task of fusing contexts should shift to a shared infrastructure which decouples applications with sensors and context fusion mechanism, provides high quality contexts, and fosters ease of programming.

* This work is partially supported by NSFC (60233010, 60273034, 60403014), 973 Program of China (2002CB312002), 863 Program of China (2005AA113160) and NSF of Jiangsu Province (BK2002203, BK2002409).

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 920–929, 2005.
© IFIP International Federation for Information Processing 2005

For solving these problems, we establish an enhanced ontology based context model using semantic web languages and design a context fusion infrastructure not only generating high-level contexts formally but also making contexts timely, accurate and consistent. With dynamic deploying of domain contexts, the infrastructure can be easily applied to different domains. Also, we provide an easy application programming model and develop a small context-aware application−Seminar Assistant. To evaluate the feasibility of our infrastructure, we give a performance study.

The rest of this paper is organized as follows. Section 2 discusses on related work. In Section 3 we present our context model. The context fusion mechanism is presented in Section 4. The infrastructure is introduced in Section 5. The performance study is shown in Section 6. Section 7 presents our application programming model and an example. Finally, we conclude in Section 8.

2 Related Work

In previous work, researchers have proposed many context models such as key-value, XML, object, UML-ER, Ontology and so on [3], and fused contexts formally or informally.

Among Systems which use informal fusion mechanism, the typical one Context-Toolkit [1] established an object-oriented framework and a number of reusable components; Cooltown [4] proposed a web-based context-aware system; Context Fabric [5] defined a Context Specification Language and a set of core services. But all of them lack a general context representation and fuse different contexts in different ways. Although ubi-UCAM [6] uses a formal unified 5W1H (Who, What, Where, When, Why and How) object-based context model, it fuses contexts informally. Because of informal context fusion, those frameworks and infrastructures are difficult to reuse.

Formal context fusion mechanism can generate high-level contexts for different applications in the same way so that the common module for fusing contexts can be extracted as a shared, highly reusable infrastructure. Karen [7] modeled contexts using both ER and UML models. Anand [8] represented contexts in Gaia system as first-order predicates written in DAML+OIL[1]. CoBrA [9] established an ontology based context model and an agent-based system for smart room environment. SOCAM [10] proposed an OWL[2] ontology based context model addressing context sharing, reasoning and knowledge reusing, and built a service oriented middleware infrastructure for applications in a smart home. However, none of them considered the lifecycle management of contexts or introduced their conflict resolution principles. And, none of the infrastructures can be easily customized for different domains. Also, they didn't give us a clear and easy application programming model.

3 Context Model

A good context model can lead to a well designed context fusion mechanism. The ideal context model which can make contexts easily shared by different applications should

[1] DAML+OIL reference: http://www.w3.org/TR/daml+oil-reference

[2] OWL reference: http://www.w3.org/TR/owl-ref

also enable context fusion both to make high-level contexts more timely, exact, complete, conflict-free and to evolve easily.

To achieve this ideal context model, our context model consists of 2 parts: ontology and its instances. The ontology which is a set of shared vocabularies of concepts and the interrelationships among these concepts enables context sharing, logic reasoning and knowledge reusing [10]. Instances of ontology include both persistent contexts and dynamic contexts. Persistent contexts usually have a long life period and they would be combined with dynamic contexts during reasoning process. Triples described as (subject, predicate, object) are used to model persistent contexts. For example, the context "Tom is a teacher" is modeled as (Tom, type, teacher). Dynamic contexts with transient characteristics only have a short life in the system, such as "Tom in Room311". Quintuples (subject, predicate, object, ttl, timestamp) are used to model them. Ttl denotes the life period of a context while timestamp means the UNIX time when the context is generated or updated.

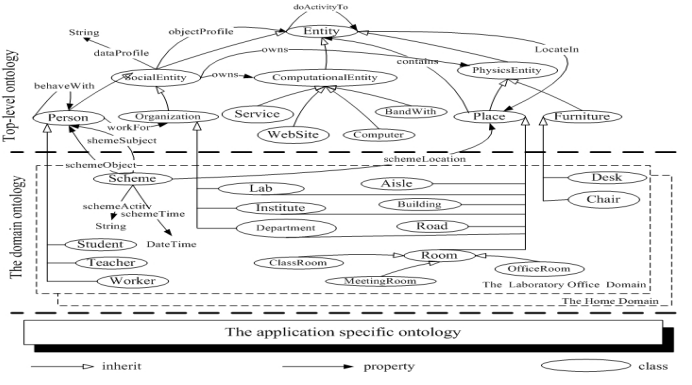


Fig. 1. Part of Our Ontology

In our implementation, the ontology is modeled by OWL-Lite. Expressive descriptions such as "TransitiveProperty", "SymmetricProperty", "Disjoint" and so on in OWL-Lite can largely decrease the number of user-defined rules. We construct our ontology in a hierarchical manner, fostering knowledge reuse. From top to down, there are top-level ontology, domain ontology and application specific ontology. Compared with CoBrA [9] which combines activities and entities as classes like "PeopleInMeeting" and SOCAM [10] which describes most activities as classes like "Talk", our ontology uses properties to represent activities for ease of describing contexts uniformly in triples or quintuples. Fig. 1 shows part of our ontology which is specialized for the laboratory office domain. Persistent contexts are serialized as RDF[3] files while dynamic contexts are implemented as RDF messages attached with ttl and timestamp.

[3] RDF reference: http://www.w3.org/TR/rdf-ref

4 Context Fusion Mechanism

Upon the context model, our context fusion mechanism is logic inference based. We apply rule based reasoning and ontology based reasoning orderly on raw contexts to generate high-level contexts, using the Jena API[4]. The user-defined rules are in the form of Jena generic rules without negation and "or" operation. For separating context generation and applications, we use deductive rules rather than reactive rules. Fostering convenient lifecycle management and conflict resolution, we modify the Jena source code and add time information to contexts during reasoning.

4.1 Adding Time Information During Reasoning

Time information is added to dynamic high-level contexts during reasoning. In detail, when a high-level context $context_{conclude}$ is generated, we have its premise context set $Facts$: $\{context_1, context_2, \ldots, context_n\}$, which has a subset $DynamicFacts$ consisting of dynamic contexts: $\{dyncontext_1, dyncontext_2, \ldots, dyncontext_k\}$. First, we select $context_{min}$ which has the minimum value of ttl plus timestamp in DynamicFacts. Then we set the ttl and timestamp of $context_{conclude}$ the same as $context_{min}$'s. The princeple here is to make the ttl and timestamp of every high-level context the same as the earliest dying one in its premise set. The reason is that there are only "and" operations to connect premises in Jena generic rules and a high-level context will become demoded when whichever of its premises is demoded. Because the operation of adding time restriction is embedded in every inference operation, there is little additional time used.

4.2 Reasoning High-Level Context

When we infer high-level contexts, rule reasoner and ontology reasoner are used orderly. The two reasoners are configured as traceable in order to facilitate conflict resolution, though much more memory is used.

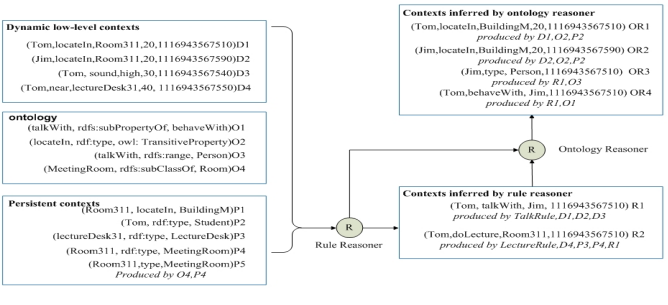


Fig. 2. An Example of Context Reasoning

[4] Jena2 Semantic Web Toolkit: http://www.hpl.hp.com/semweb/jena2.htm

Here we take an example to explain the reasoning process. As Fig. 2 shows, in the example, we have 4 dynamic low-level contexts, 3 (D_1, D_2, D_4) of them coming from Cricket[5] related context provider while another (D_3) coming from Mica sensor[6] related context provider. Also, 4 assertions (O_1, O_2, O_3, O_4) in ontology and 4 persistent contexts (P_1, P_2, P_3, P_4) have been deployed to the platform. But persistent context P_5 is deduced from O_4 and P_4 when (P_1, P_2, P_3, P_4) are being deployed to the platform. The two user-defined rules used in this reasoning instance are shown as follows:

TalkRule: $(?x \quad locateIn \quad ?room), (?y \quad locateIn \quad ?room), (?room \quad rdf: type \quad Room), (?x \quad sound \quad high) -> (?x \quad talkWith \quad ?y)$
LectureRule: $(?x \quad locateIn \quad ?room), (?room \quad rdf: type \quad MeetingRoom), (?x \quad near \quad ?lectern), (?lectern \quad rdf: type \quad Lectern), (?x \quad talkWith \quad ?y) -> (?x \quad doLecture \quad ?room)$

We can see that after reasoning, high-level contexts $\{R_1, R_2, OR_1, OR_2, OR_3, OR_4\}$ are generated.

4.3 Context Lifecycle Management

With time information for contexts, a lifecycle management is brought in for dynamic contexts. Fig. 3 shows context lifecycle.

A background thread runs periodically to tick the life period for every live context. When a context's ttl is no more than zero, it is demoded and removed to historical context storage. We store demoded contexts in persistent storage rather than discard them because historical contexts may be useful for various applications. When a new generated context has the same (subject, predicate, object) as an existing one, we update the older one by making the newer displace it. Applications can register callbacks not only by specifying interested contexts but also by pointing out which transition (generated, updated or demoded) they want.

Through lifecycle management, demoded contexts have been removed away so that context conflicts are fewer than normal. The contexts for applications can be more accurate and timely.

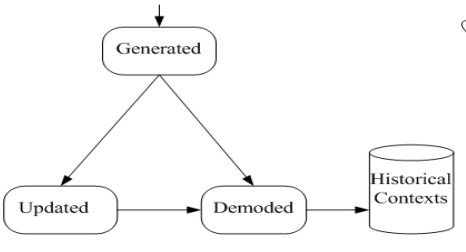


Fig. 3. Lifecycle of Dynamic Contexts

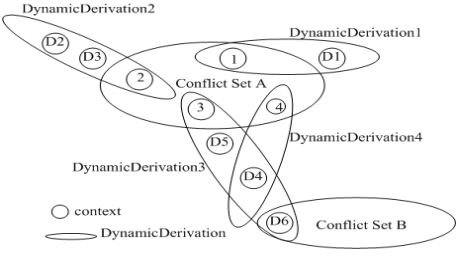


Fig. 4. An Example of Conflict Resolution Algorithm

[5] The Cricket indoor location system: http://cricket.csail.mit.edu/
[6] The Mica Sensor: http://www.xbow.com

4.4 Conflict Resolution

Inconsistent contexts are often emerging in systems due to faults of either physical sensors or software. Using ontology can facilitate conflict detection. For example, if our context repository has 2 dynamic contexts: (Tom, locateIn, Room311, 15, 1116943567511) and (Tom, locateIn, Aisle3, 15, 1116943567599), 2 persistent contexts: (Room311, rdf:type, Room) and (Aisle3, rdf:type, Aisle), and 2 assertions in ontology: (Room, owl: disjointWith, Aisle) and (locateIn, rdf:type, owl:FunctionalProperty), a conflict will be detected in ontology model because there is an instance of both Room and Aisle. But how to resolve this conflict and make contexts consistent? We've designed an algorithm to resolve context conflicts, which is based on the time factors of contexts. Our design principle is that the later contexts and persistent contexts have more priorities.

From the Jena API, a validity report can be obtained, which indicates every first-hand conflicting pairs. Then we can easily construct several conflict sets whichever consists of contexts conflicting with each other, from the validity report. For every context $conflictcontext_i$ in every conflict set, we trace its derivation and make a set called $DynamicDerivation_i$ which contains dynamic contexts in $conflictcontext_i$'s derivation set and $conflictcontext_i$ itself. For the example in Fig. 2, R_2(Tom, doLecture, Room311, 1116943567510) is a high-level context and its $DynamicDerivation$ is $\{D_1, D_2, D_3, D_4, R_1, R_2\}$. Then we resolve conflicts for every conflict set orderly.

Now we explain in detail how to resolve conflict for one conflict set called $Conflicts$. We compare each context in the $Conflicts$. If two dynamic contexts conflict, we discard the earlier one's DynamicDerivation. But during discarding a DynamicDerivation, contexts which exist in other existing DynamicDerivations are reserved. As an exception, if there is any persistent context in $Conflicts$, all other contexts' DynamicDerivation should be discarded. It is ensured that there is no conflict among persistent contexts because when they are deployed to our infrastructure, we check the consistency and reject conflicting ones.

Fig. 4 shows an example of the conflict resolution algorithm. In the example, we have two conflict sets: conflict set A and conflict set B. We first resolve conflicts for A, and then for B. Assume that in A, $context_1$, $context_2$, $context_3$ and $context_4$ are ordered by their timestamp increasingly. After A is resolved by the algorithm, there are 3 contexts$-context_4$, $context_{D4}$, $context_{D6}$ left.

It is obvious that through the conflict resolution, there is no conflict left in context set although some right contexts may be discarded. Nevertheless, contexts are different from general knowledge in the way that correct contexts are generated frequently while wrong contexts emerge by accident. Therefore, correct contexts will appear soon even if they are wrongly deleted while the probability of wrong contexts' repetitious appearances is very low. However, the conflict resolution algorithm is a computational intensive task so that we make it run periodically.

5 The Middleware Infrastructure

Based on this context fusion mechanism, we've build a centralized OSGi[7] based context service as an infrastructure to accept raw contexts from context providers, produce high-

[7] OSGi, Open Service Gateway Initiative: http://www.osgi.org

level contexts, perform context lifecycle management, resolve context conflicts, and deliver contexts to all applications. In the system, besides the context service, there are following roles: sensors which gather data from the physical world, such as location, temperature, pressure, picture, noise and so on; context providers which interact with sensors, and transform sensor data into raw semantic contexts; and applications (context consumers) which consume contexts by either subscribing or querying, and adapt to context changes.

In our smart environment, sensors, context providers, context consumers and the central context service are physical distributed. Computers connect through a local network. To ease the communication among different nodes, we use Java RMI technology. One feature of the infrastructure is dynamic deploying of ontology, persistent contexts and rules without restarting the service. With the help of domain independence of the context service, application developers of our infrastructure can easily customize the context service to their specific domains by deploying the domain related ontology, persistent contexts and rules. In our prototype, we customize the infrastructure to support the laboratory office domain.

6 Evaluations of Context Service

For evaluating the context service, we've tested it in following aspects:

Performance of conflict resolution. The experiment has been conducted on a set of Linux workstations with different hardware configurations (512MB-RAM/P4-1.4 GHz, 1GB-RAM/P4-2.8 GHz, and 4G-RAM/2-XEON-CPU). The ontology reasoner we have used is entailed by OWL-Lite, and the rule reasoner applies a rule set consists of 8 forward-chaining rules. Fig. 5 shows the results. The performance of our conflict resolution algorithm is acceptable when a high-powered workstation is used.

Effect of conflict resolution. We use a simulating context provider in a client node which repeats sending two conflicting raw contexts orderly to the context service. In this way, many inconsistent contexts will occur in the system so that the consistency and accuracy of contexts can be tested. It is obvious that the interval between the two contrary contexts can largely influence the results so that we make the interval change from 10s to 40s at the step of 5s and maintain each interval for 10 minutes to carry out the test. An application in another client node queries contexts 10 times in every minute to see the probability of getting conflicting contexts and getting correct contexts for each interval. The central server is located at a Linux workstation which has 4G RAM and 2 Xeon CPUs. The experiment results are shown in Fig. 6 ("CR" means conflict resolution). We can see that with the time based conflict resolution, the quality of contexts is much better.

Performance of the infrastructure. The configuration of the experiment includes one server and two clients. The central server is also the Linux workstation with 4G RAM and 2 Xeon CPUs. A simulating context provider is running at one client node and sends raw contexts to the central server at different frequencies while an application is running at another client node and queries contexts from the central server twice a minute. Tests are carried out respectively with the number of total live contexts at 3

levels: 3000−4000(suited for a smart home), 5000−6000(suited for a small office) and 8000−9000(suited for a middle office). We record the latency of this application for each frequency within every total context range. From the results shown in Fig. 7, we can see that after a curve point in each line, the latency of acquiring contexts increases in an exponential speed.

From the results of our experiments, we can get several useful guides for design context fusion mechanism:

Firstly, our time based conflict resolution algorithm is an expensive task so that it is impossible to run it every time when high-level contexts are generated. But it is necessary to run the algorithm periodically to decrease the probability of conflicts and faults.

Secondly, this logic inference based context fusion mechanism isn't suitable for real-time applications. But it's still valuable for non-time-critical applications in domains such as office, home, hotel, school and so on.

Thirdly, the centralized design with a bottleneck existing isn't suitable for large smart space so that it is necessary to develop distributed context fusion mechanisms.

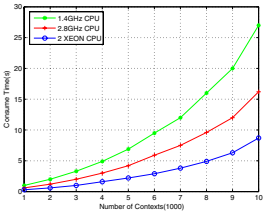


Fig. 5. Performance of Conflict Resolution

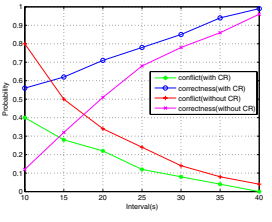


Fig. 6. Probability of Conflicts and Correctness

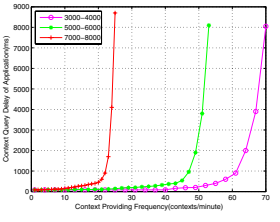


Fig. 7. Performance of The Infrastructure

7 Application Programming Model

7.1 How to Get Needed Context?

For querying contexts, we provide a RDQL[8] based context query mechanism. RDQL is a query language which can select matched RDF triples from a triple set. Applications based on our context service infrastructure can query contexts by specifying a RDQL query sentence.

Because the syntax of RDQL is complex, we use a simple example to illustrate how it works rather than introduce all of it. For example, there are 4 contexts in our context repository: A(Tom, type, teacher), B(Tom, doLecture, Room311,15, 1116943567511), C(Bob, locateIn, Room311, 15, 1116943567580) and D(Bob, type, Student). If we use a RDQL query sentence like that: "select ?x where (?x type Teacher),(?x locateIn Room311)", we can get context A and B as the result. In this way, we find that RDQL can specify very complex contexts flexibly so that it is very suitable to be a context

[8] RDQL tutorial: http://jena.sourceforge.net/tutorial/RDQL/index.html

query language. Also, applications can get historical contexts by setting the time constrains in queries.

Based on this query mechanism, a callback interface is easily designed. Applications can register their interested contexts with wanted transitions (Generated, Updated or Demoded) to the context service at central server. If registered contexts occur and their transitions are suited, the applications will be notified.

7.2 Case Study

Scenario. In research groups, seminars are often held. When someone gives a lecture, he/she should copy the slides to his/her flash disk, carry it to the meeting room, copy the slides to the computer in the meeting room, and then open them. The work is dull and trivial, and many of people's attentions are consumed. In our context-aware computing environment, the lecturer needs to do nothing other than edit his/her lecture notes. When he/she enters the meeting room, and stands near the lectern, his/her slides will be opened automatically. During the seminar, if some strangers come in, a warning balloon will pop up on the screen. At the end of the seminar, the slides will be closed automatically.

Implementation. We implement the scenario based on the context service. The application called Seminar Assistant has two parts. One called User Assistant runs at all users' computers while the other called Meeting Assistant runs at the computer in the meeting room. When the User Assistant detects the context that the user it serves will give a lecture in the next few days, it will upload the slides he has edited recently, the name of which matches the lecture to an http server. When the lecturer starts to give the lecture in the meeting room, the Meeting Assistant will obtain the right context, and then download and open the previous uploaded slides. Then the Meeting Assistant starts detecting if strangers come in. When the Meeting Assistant detects the context that the lecturer leaves the room, it will close the slides. In this application, we've used the in-door location sensor Cricket to detect a person's location in a room, and also the Mica sensor to detect the noise in a room. Fig. 8 shows a piece of the source code.

```
                    Seminar.java
public class Seminar implements ContextConsumer {
   ......
   public void callBackOnContexts(Iterator contexts) {
          Context context = (Context)contexts.next(); //get context
      String lecturer = context.getSubject().getName();
          openLectureNote(lecturer); //Open the Slides
   }
   public void register()
   {
     String query = "select ?x where (?x doLecture "+
          peoperties.getProperty("This-Room")+")";
        ContextFilter
          filter=contextFactory.createQLContextFilter(query,GENERATED)
          ;
     contextService.registerContextConsumer(this,filter);
     //register interested context to the ContextService
      }
...... }
```

Fig. 8. A Piece of The Code of Seminar Assistant

Fig. 9. The Runtime Effect of Seminar Assistant

Runtime Effect. Fig. 9 shows the runtime action of Seminar Assistant when a stranger comes into the meeting room during a seminar. Part of the context reasoning process for this example is already shown in Fig. 2.

8 Conclusion and Future Work

Our study in this paper shows that our context fusion mechanism is feasible and necessary for providing context-aware applications with high quality contexts. We've implemented a context fusion service as an infrastructure to support context-aware applications, decoupling applications with sensors and context fusion. A case study shows that it's easy and rapid to develop applications based on our platform. The work of this paper is part of our ongoing research project—FollowMe which aims at a workflow-driven, service-oriented, pluggable and programmable context-aware infrastructure. Our next step is to explore novel approaches to both improve the quality of contexts and reduce the time cost. Also, we are working towards a distributed context processing mechanism to make context-aware systems more efficient and robust.

References

1. A. K. Dey, D. Salber, G. D. Abowd. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. Anchor article of a special issue on Context-Aware Computing, Human-Computer Interaction (HCI) Journal, 16(2-4), 2001, 97-166.
2. Weiser. M. Computer for the 21st century. Scientific American, 265(3), 1991, 94-104.
3. T. Strang, C. Linnhoff-Popien. A Context Modeling Survey. Workshop on Advanced Context Modelling, Reasoning and Management as part of UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England, September 2004.
4. T. Kindberg, J. Barton. A Web-based Nomadic Computing System. Journal of Computer Networks. 35(4). 2001: 443-456.
5. J. Hong and J. Landay. An infrastructure approach to context-aware computing. Human Computer Interaction (HCI) Journal, 16(2-4), 2001: 287-303.
6. S. Jang, W. Woo. Ubi-UCAM: A Unified Context-Aware Application Model. In Proceeding of Modeling and Using Context – the 4th International and Interdisciplinary Conference. Springer. Stanford, CA, USA. June 2003. pp.178-189.
7. Karen Henricksen, et al. Modeling Context Information in Pervasive Computing Systems. First International Conference on Pervasive Computing. Springer. August 2002. Zrich, Switzerland. pp.167-180.
8. Anand Ranganathan, et al. A Middleware for Context-Aware Agents in Ubiquitous Computing Environmentse. In Proceeding of ACM/IFIP/USENIX International Middleware Conference. Springer. June 2003. Rio de Janeiro, Brazil. pp.143-161.
9. H. Chen, T. W. Finin, A. Joshi, L. Kagal, F. Perich, D. Chakraborty. Intelligent Agents Meet the Semantic Web in Smart Spaces. IEEE Internet Computing, (November 2004):69-79.
10. T. Gu, H. K. Pung, D. Q. Zhang. Towards an OSGi-Based Infrastructure for Context-Aware Applications in Smart Homes, IEEE Pervasive Computing, 3(4) 2004, 66-74.

A Framework for Video Streaming to Resource-Constrained Terminals

Dmitri Jarnikov[1], Johan Lukkien[1], and Peter van der Stok[2]

[1] Dept. of Mathematics and Computer Science, Eindhoven University of Technology,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
{d.s.jarnikov, j.j.lukkien}@tue.nl
[2] Philips Research Prof Holstlaan 4,
5656 AA Eindhoven, The Netherlands
peter.van.der.stok@philips.com

Abstract. A large range of devices (from PDA to high-end TV) can receive and decode digital video. However, the capacity of one device is very different from the capacity of another device. This paper discusses how the same video but differently encoded can be efficiently distributed to a set of receiving devices. The key technology is scalable video coding. The paper shows how a framework assists in adapting the digital code to the changing transmission conditions to optimize the quality rendered at the different devices. The paper concludes with a validation based on a real-time streaming application.

1 Introduction

Delivering a high quality video over a network and seamless processing of the video on various devices depend for a larger part on how the network and device resources are handled. The paper addresses a concurrent video content distribution to multiple resource-constrained devices, focusing on wireless in-home networks. Fig. 1 shows a schematic view of a simple video transmission system that consists of a set of terminals wirelessly connected to a sender. A sender is a powerful device that stores videos and provides media access to video data for all receivers in the net.

Terminals, being mostly CE devices are resource-constrained. Moreover, the devices have different processor/memory capabilities, thus not every terminal may be capable of processing all video data that is streamed by the sender. To ensure that video data is processed by all terminals, a sender should send to each terminal the amount of data that can be successfully processed by the terminal. This is usually done by performing a content adaptation at the sender.

There are several strategies for content adaptation. The three foremost of those strategies are the simulcast model, transcoding and scalable video coding model. With the simulcast approach the sender produces several independently encoded copies of the same video content at varying features, such as different bit/frame rates, and spatial resolutions. The sender delivers these copies to the terminals, in agreement with the specifications coming from the terminal.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 930 – 939, 2005.
© IFIP International Federation for Information Processing 2005

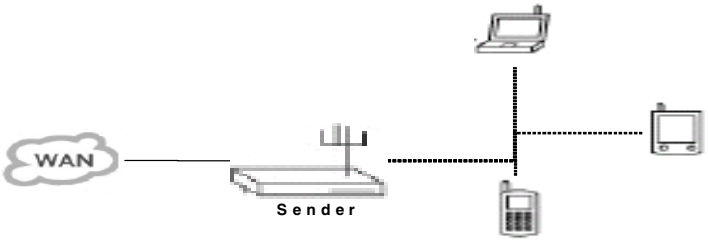


Fig. 1. General view of the system

Transcoding of a video stream is converting the video stream from one format to another (e.g. from MPEG-2 [3] to MPEG-4 [4]) or transformation of the video stream within the same media format (e.g. change of frame rate, bit-rate or image resolution).

A scalable video coding scheme describes an encoding of video into multiple layers, including a Base Layer (BL) of basic quality and several Enhancement Layers (EL) containing increasingly more video data to enhance the quality of the base layer [9]. Scalable video coding is represented by variety of methods that could be applied to many existing video coding standards [3,4,5]. These methods are based on principles of temporal, signal-to-noise ratio (SNR) and spatial scalability [7]. Combinations of the scalability techniques produce hybrid scalabilities (e.g. spatial-temporal).

In a heterogeneous environment there are terminals of different types, so multiple possibilities for content adaptation are possible. We propose a framework that enables video streaming to terminals with different resource constraints (the rest of the paper addresses only processing resources) based on a uniform content adaptation for all terminals.

2 Analysis

We distinguish three types of terminals. The first type has a fixed decoding process, based on a standard legacy decoder. The second type has a decoding process that is organized as a scalable video algorithm (SVA) [8], which allows changing of the quality of the processing to trade-off resource usage for output quality of the algorithm. The third type of terminals is capable of processing data that is created with scalable video coding techniques.

Type 1. Most terminals are equipped with a standard software or hardware decoder. In this case simulcast is the simplest content adaptation method. Every terminal in the system requests a stream that fits closely to the current resource limitations of the terminal. The overall bandwidth consumption of the simulcast approach is a sum of the bit-rates of all streams sent to the terminals. So, the simulcast strategy fills up the network bandwidth with a number of variations of the same content, which results in over utilization of the bandwidth. Also, if the bandwidth has frequent fluctuations, which is often the case in wireless networks, the simulcast approach is not the best technique because when available network bandwidth is lower than overall bit-rate of all streams, some of the streams need to be cut off.

Transcoding is another content adaptation technique for this type of terminals. The size of the video stream *BT* that can be processed by all terminals is chosen such that it satisfies the requirements of the weakest terminal

$$BT = \min(B_i) \text{ for } i = 1..N , \tag{1}$$

where B_i is the highest bit-rate that can be processed by receiver i.

Network bandwidth fluctuations could be handled by lowering/raising video stream bit-rate. Evidently, the highest possible bit-rate, *BA*, for the stream is defined as a minimum between maximum bit-rate allowed by network, *BN*, and suitable for all terminals

$$BA = \min(BN, BT) , \tag{2}$$

where BN is the currently available network bandwidth. A disadvantage of the approach is that receiver i in the system has unused resources

$$U_i = B_i - \min(B_j) \text{ for all } j = 1..N . \tag{3}$$

Type 2. If receivers are capable of adjusting the quality of decoding process, i.e. change usage of resources for decoding, the utilization of terminal and network resources can be improved. Each terminal can be characterized by the highest stream bit-rate (B) that can be processed. If resources are not sufficient for the complete processing of the input data with bit-rate B, the processing is performed with a lower quality. So, the output quality of the decoder lowers to a given level Q, which is the maximal quality that can be achieved under current resource limitations. We define BM as a bit-rate that can be fully processed by the terminal with the highest quality of the processing and produces the same quality Q with the current resources allowance.

In the system, a video stream is transcoded to the bit-rate that satisfies the requirements of the strongest terminal, but still can be processed by all others

$$BT = \min\left[\max(BM_i), \min(B_i)\right] \text{ for } i = 1..N . \tag{4}$$

The highest possible bit-rate is calculated as in (2).

The transcoding approach has difficulties in handling fast bandwidth changes. Whenever a change occurs, the transcoder adapts the bit-rate of the video stream. However, some time is necessary to detect the change and communicate it to the transcoder. Additional time is needed for the transcoder to adapt to new settings.

Type 3. If terminals are able to process scalable video coded streams, a scalable video coding adaptation technique is an effective solution to the drawbacks mentioned above. The advantage of scalable video is the easy adaptation to varying channel conditions. The content adaptation can be done on a per frame basis. This is very important for wireless video transmission, where a bandwidth change is fast and unpredictable. Also, usage of scalable video coding gives a terminal the possibility to subscribe to as many layers as it can handle.

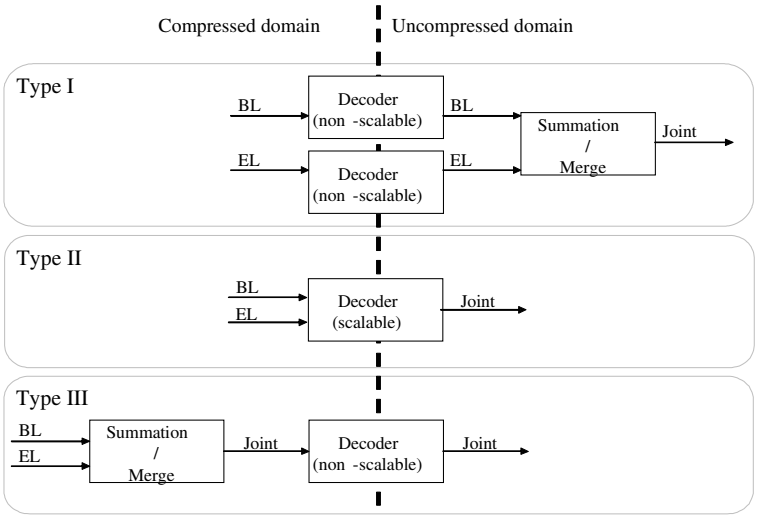


Fig. 2. Three types of organizing decoding process for scalable video coding

Possible types of organization of the decoding process for scalable video are depicted in Fig. 2. Type I assumes a video where BL and EL are encoded as a standard non-scalable video streams. The streams are decoded by different decoders and then merged together. It is possible to decide upfront, how many layers will be processed. Type II takes scalable video that complies with a standard scalable technique. The output of the decoder is a joint decoded video. Type III operates with video streams that can be merged together before being decoded.

The schemes shown in Fig. 2 provide a higher quality at the cost of increased resource consumption. The whole decoding process can be seen as a SVA. Choosing the number of processed layers is the key to trade-off quality for resource consumption. A decoding process may drop processing an enhancement layer at any moment thus providing complete input of a frame at the best quality possible given the processing limitations. Processor consumption can be changed on a per-frame basis in real-time.

An important requirement for a scalable video coding technique is that successful transmission and decoding of the BL should be guaranteed. This is not possible if the available bandwidth drops below BL bit-rate or there are not enough resources to decode BL. To handle this situation, a reconfiguration of layers (i.e. changing of number of layers and their bit-rates) at run-time is proposed.

The characteristics of the terminal, which processes scalable video, are the maximal number of layers (NL_i) and the maximal bit-rate of the BL (B_i). The bottleneck of the approach is the complexity of choosing a layers configuration that suits both network and terminals resources. Although it is easy to calculate the highest possible bit-rate for BL as in (2), the number and bit-rates of EL are usually chosen manually.

Conclusion. In general, transcoding into multiple layers is suitable for use with all types of terminals, where a one layered transcoding is a particular case of scalable

video coding with *NL* equal to 1. We suggest a scalable video coding as a uniform content adaptation technique. In this paper we propose a general framework for video streaming to multiple terminals that is based on scalable video coding.

3 Framework Description

The design of our framework is depicted on Fig. 3. A receiver (terminal wirelessly connected to a sender) processes incoming video data in accordance with local resource limitations. The sender makes an adaptation of input video data in conformity with requirements of the terminals and the network conditions. The intent of the system is that scalable-enabled receivers and non-scalable receivers use the same basic functionality and a single adaptation technique is performed based on requirements of all terminals.

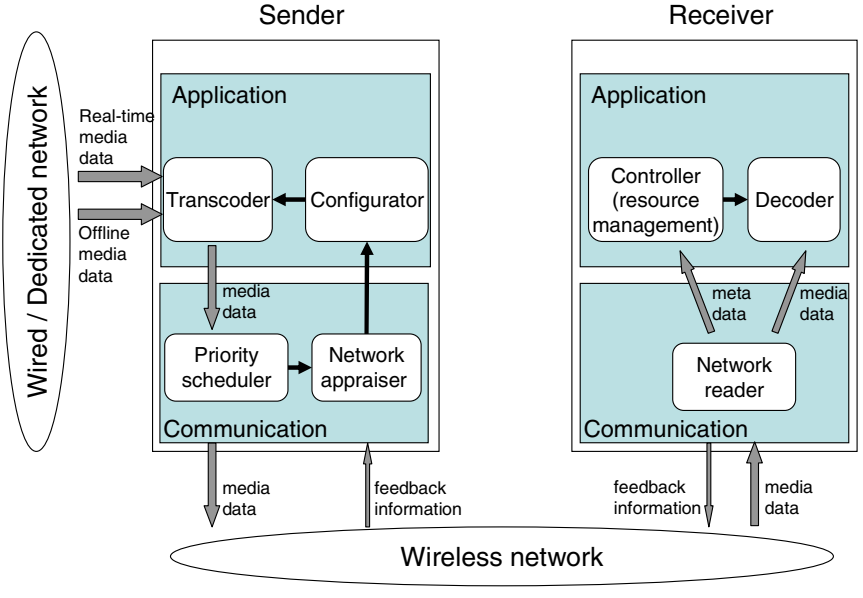


Fig. 3. Design of the system

3.1 Receiver

The receiver is responsible for receiving the required number of layers from the offered ones, decoding, and displaying video data. Also, it reports to the sender the maximal number of layers (*NL*), maximal bit-rate that can be processed without lowering processing quality (*BM*) and maximal bit-rate *B* . Additionally, the receiver provides a feedback to the sender in relation to the network statistics (how many layers/frames where successfully received). The three components shown in the receiver in Fig. 3 are discussed in detail.

Network Reader. The main task of the network reader is to pre-process video data. In the case of scalable video coding, transmission delays may lead to loss of synchronization between the layers. Frames from both base and enhancement layers should be merged during decoding process. If a frame from enhancement layer comes too late, this frame is discarded before offering it to the decoder. The network reader informs the local controller about the amount of received video data. Network reader communicates network statistics back to the sender.

Decoder. Unless the terminal is equipped with a standard decoder, the decoding process is implemented as a scalable video algorithm. The amount of processed data is a parameter that changes the output quality as well as the resource consumption.

A decoding process for scalable video coding drops an enhancement layer to fit into the given processing resources. The control over resource usage can be changed accordingly.

Controller. A decoder implemented as a SVA needs a controller to assure that resource consumption of the decoder is scaled to meet the current limitations. However, changes in the amount of processed data result in a picture quality change. Frequent changes of picture quality are not appreciated by user. A control strategy, as proposed in [6,1], minimizes the number of quality changes while maximizing the average output quality.

3.2 Sender

The sender takes online or offline content and transcodes it into a scalable video. This video is sent over a wireless network to multiple receivers. A feedback from the receivers is used for making a decision about configurations of the scalable video, i.e. choosing number and bit-rates of layers.

Scalable Video Transcoder. The transcoder converts non-scalable video into multi-layered scalable video in accordance to the current layer configuration. The configuration may be changed at run-time. The input to the transcoder is provided via a reliable channel, thus assuming that there are no losses or delays in the incoming stream.

An important requirement for the transcoder is the ability to incorporate information about currently chosen configuration into data streams. Writing configuration settings into the base layer allows natural propagation of changes through the system, as all involved parties may be informed on the complete change history.

The encoded streams must have two properties to allow dropping of frames of a layer without consequences for other frames: 1) no dependencies of base layer frames on enhancement layer frames [2], 2) frames in an enhancement layer should have no relation to each other.

Priority Scheduler. If transcoder outputs ELs along with BL, the sender uses a priority scheduler, which ensures that layers of a scalable video are sent in accordance to their priority. Since BL information is absolutely necessary to enable scalable video usage, this layer has the highest priority. The priority of EL decreases with increasing layer number.

When a channel degrades, the buffers of the sender get full. This affects the low priority streams on the first place and propagates towards the higher priority streams. The information regarding the fullness of the sender buffers is communicated to the network appraiser as an indication of the network status.

Network appraiser. The information from the scheduler is a rough but fast indication of a network condition change. In turn, a feedback channel delivers more precise information expressed in terms of error rate and burstiness. The network appraiser collects information from the priority scheduler and from receivers and communicates changes to the layer configurator.

Layer configurator. The layer configurator chooses number and bit-rates of layers based on the acquired information about network conditions and terminals status. The network information is used to estimate BN (currently available network bandwidth), worst-case error rate and burtiness of the errors. The terminal information is used to define the maximal number of layers that could be produced

$$L = \max(NL_i) \text{ for } i = 1..N \tag{5}$$

and the required BL bit-rate that is calculated based on (4). The layer configurator uses a pre-calculated strategy to choose a suitable configuration based on the above-mentioned values. A strategy is created offline by a network simulation environment.

Strategies are created with knowledge of maximal number of layers and the currently available network bandwidth. The maximal bit-rate for BL is not taken into account, as it again increases the number of needed strategies. If a lower BL bit-rate should be chosen at run-time due to terminals requirements, the bit-rate of BL is re-distributed to the first EL.

4 Evaluation

A prototype has been constructed to validate the approach described in this paper. Our implementation of the scalable video transcoder is based on a modified MPEG-2 SNR video coding and corresponds to the decoding type I (Fig. 2), which suggests standard non-scalable decoders with an external summation of streams after the decoders. The implementation of our coding overcomes an important weakness of a standardized approach [3]: the dependency of BL on EL. Moreover, our scheme allows any number of ELs, which provides greater flexibility for accommodation to various network conditions. The details of the implementation are presented in [2].

The resource management is done based on the network aware controller [6]. The approach is based on a Markov Decision Process (MDP). The MDP is solved offline and the calculated optimal control strategy is applied online to control the decoding algorithm [6].

For the evaluation we used a system with three different types of terminals. Terminal 1 is equipped with standard MPEG-2 decoder and is capable of processing at run-time standard definition streams of not more than 3.5 Mbps. Terminal 2 is capable of adjusting the quality of decoding process. It handles standard definition streams of up

to 6 Mbps, however only 4 Mbps could be efficiently decoded at run-time. Terminal 3 handles scalable video consisting of at most 3 layers and BL bit-rate should not exceed 8 Mbps.

Fig. 4 shows an example of layer configurations for different system setups. If Terminal 1 is the only receiver in the system, the sender streams only one video stream with a bit-rate not higher than 3.5 Mbps. If network conditions do not allow successful transmission of video, the bit-rate is decreased to fit into present network limitations. A similar behavior is observed for Terminal 2 as a receiver, the only difference is that bit-rate increases up to 4 Mbps (maximal efficiently decoded bit-rate), whenever network conditions allow it. For the system with Terminal 3, the configurations with BL and one EL are chosen as most optimal. New configuration of BL and EL is chosen when the network conditions change.

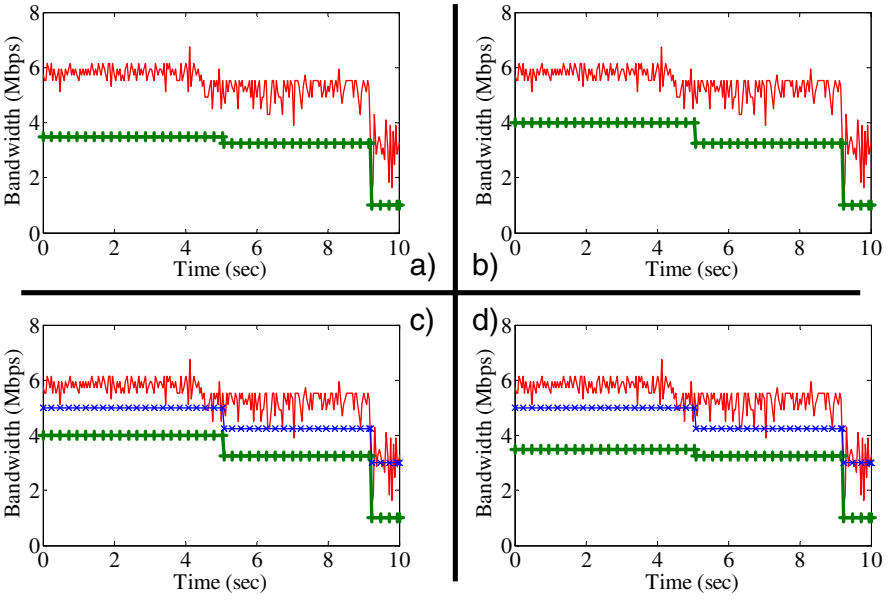


Fig. 4. Configuration of layers for different terminals: a) Terminal 1; b) Terminal 2; c) Terminal 3; d) Terminals 1, 2 and 3. Top line is available bandwidth, crossed (+) bottom line is BL bit-rate, crossed (x) line is bit-rate of BL and EL.

Bringing all three terminals together changes the configuration of layers (Fig. 4, d). The layer configuration delivers a BL and EL, with Terminals 1 and 2 subscribing only to BL and Terminal 3 receiving both layers. However, the BL bit-rate is chosen based on the requirements of the weakest terminal, which is Terminal 1. During time interval [0, 4.5] the BL bit-rate is lower than the optimal for Terminal 3 (Fig. 4, c). Because the optimal BL bit-rate of 4 Mbps is higher than the maximal bit-rate that Terminal 1 can handle (3.5 Mbps). The difference of 0.5 Mbps is reassigned to EL, resulting in the same overall bit-rate for layers.

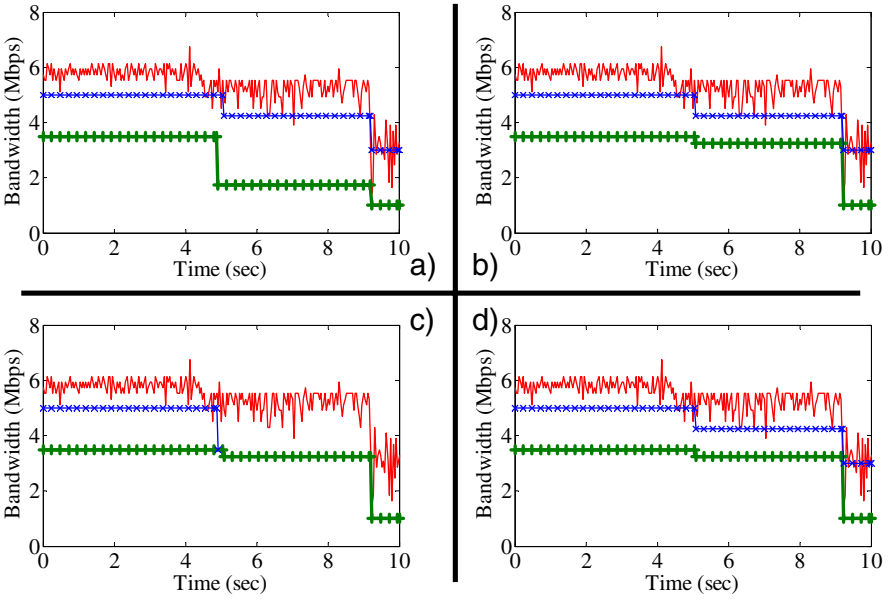


Fig. 5. Configuration of layers under changing resources availability: a) change on Terminal 1; b) change on Terminal 2; c) change on Terminal 3; d) no changes. Top line is available bandwidth, crossed (+) bottom line is BL bit-rate, crossed (x) line is bit-rate of BL and EL.

As the second step we looked at the setup with all three terminals. We simulate starting of another application on a terminal by lowering its processing capabilities by half. Fig. 5 shows an example of changed layer configurations because of resource availability change on one of the terminals. The change occurs 5 seconds after start.

If Terminal 1 has a drop in available resources, it results in lower BL (Fig. 5, a). For scalable video the drop in BL bit-rate is compensated by increasing the bit-rate of EL such that the total bit-rate of the layers is the same as for the optimal configuration (Fig. 5, d). If a resource drop occurs in Terminal 2, it has no influence on layer configuration. The BL size stays the same and Terminal 2 lowers quality of processing to fit into current resource allowance. Finally, if Terminal 3 experiences a shortage of resources, it drops processing of EL.

5 Conclusions

In this paper we presented a framework for achieving high quality video transmission over wireless networks based on scalable video coding. Important quantities are the number of layers and the size of the layers. A change of the network conditions forces a change to the number or size of one or all layers. The paper shows at which moments such changes need to be initiated and what their value should be.

The paper also shows that the same scheme still applies when the receiving decoder can only handle one layer. Consequently, the proposed scalable video based content adaptation technique can be applied generally for all types of terminals.

References

1. C.C.Wust, L.Steffens,R.J.Bril, and W.F.J.Verhaegh, "QoS Control Strategies for High Quality", Video Processing. In Proc. 16th Euromicro Conference on Real-Time Systems (ECRTS), Catania, Italy, 2004.
2. Dmitri Jarnikov, Peter van der Stok, Johan Lukkien, "Wireless streaming based on a scalability scheme using legacy MPEG-2 decoders", Proceedings of The 9th IASTED International Conference on Internet and Multimedia Systems and Applications, ACTA Press, 2005.
3. ISO/IEC International Standard 13818-2, "Generic Coding of Moving Pictures and Associated Audio Information: Video", Nov., 1994.
4. ISO/IEC International Standard 14496-2, "Information Technology – Generic Coding of Audio-Visual Objects, Part 2: Visual", MPEG98/N2502a, Oct., 1998.
5. ITU-T International Telecommunication Union, "Draft ITU-T Recommendation H.263 (Video Coding for Low Bit Rate Communication)", KPN Research, The Netherlands, Jan., 1995.
6. Jarnikov, D.; van der Stok, P.; Wust, C.C., "Predictive Control of Video Quality under Fluctuating Bandwidth Conditions". Multimedia and Expo, 2004. ICME '04. 2004 IEEE International Conference on, Volume: 2 , pp. 1051 – 1054, June 27-30, 2004
7. McCanne, S., Vetterli, M., Jacobson, V., "Low-complexity video coding for receiver-driven layered multicast", IEEE journal on selected areas in communications, vol. 16, no 6, p.983-1001, 1997.
8. R.J. Bril, C. Hentschel, E.F.M. Steffens, M. Gabrani, G.C. van Loo and J.H.A. Gelissen, "Multimedia QoS in consumer terminals", Proc. IEEE Workshop on Signal Processing Systems (SIPS), pp. 332-343, Sep. 2001.
9. Yao Wang, Joern Ostermann, and Ya-Qin Zhang, "Video Processing and Communications", Prentice Hall, 2002

Fragile Watermarking Scheme for Accepting Image Compression

Mi-Ae Kim, Kil-Sang Yoo, and Won-Hyung Lee

Department of Image Engineering,
Graduate School of Advanced Imaging Science,
Multimedia & Film, Chung-Ang University, #10112,
Art Center, 221 Hukseok-Dong, Dongjak-Gu, Seoul, Korea, 156-756
kimma@dreamwiz.com, lucky@ms.cau.ac.kr, whlee@cau.ac.kr

Abstract. As images are commonly transmitted or stored in compressed form such as JPEG lossy compression, image authentication demands techniques that can distinguish incidental modifications (e.g., compression) from malicious ones. In this paper, we propose an effective technique for image authentication which can prevent malicious manipulations but allow JPEG compression. An image is divided into blocks in the spatial domain, each block is divided into two parts by randomly selecting pixels, and average gray values for the parts are calculated. The average value is compared with that of the adjoining block to obtain an authentication signature. The extracted authentication information becomes the fragile watermark to be inserted into the image's frequency domain DCT block. The experimental results show that this is an effective technique of image authentication.

1 Introduction

Image authentication plays an extremely important role in the digital age, allowing verification of the originality of an image. This is due to the fact that powerful and easy-to-use image manipulation has made it possible to modify digital images without leaving evidence of modification. In order to better utilize the bandwidth and to minimize the space required for storage, most multimedia content such as images, audio or video are stored or transmitted in compressed formats like JPEG lossy compression. Consequently, an image authentication system must accept compressed images while detecting malicious manipulations such as replacement or removal of original objects.

Image authentication techniques fall into two broad categories: digital signature and digital watermark. A digital signature is based upon the idea of public key encryption. An extracted image digest is encoded using a hashing function and then transmits them to a receiver along with image. A private key is used to encrypt a hashed version of the image. If the hash values correspond, the image is authenticated. This approach does not permit even a single bit change. Therefore, it is not appropriate to apply this method to an image authentication system, as images must often be compressed and/or quality enhanced. Different from digesting of data as described above, there is the digital signature approach, which is based on the features of an image [1-4]. In this approach, which is used frequently for image authentication,

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 940 – 946, 2005.
© IFIP International Federation for Information Processing 2005

the features of an image that are resistant to common image processing (including compression) are extracted and are used as a digital signature. The digital signature is stored (or transmitted) separately from the image. Thus, the original image is not modified; however, it is cumbersome to manage digital signature separately from images. In the watermark-based approach, authentication information is inserted imperceptibly into the original image [5-8]. If the image is manipulated, it should be possible to detect the tampered area through the fragility of the hidden authentication information (watermark). Ideally, the embedded watermark should only be disrupted by malicious modifications; it should survive acceptable modifications such as compression.

Bhattacharjee and Kutter [3] proposed techniques to generate digital signatures based on the locations of feature points. The advantage of this technique is its compact signature length. However, the selection process and relevance of the selected points are unclear [10].

The scheme proposed by Chun-Shin Lu and Hong-Yuan Mark Liao [4] relies on the fact that the interscale relationship is difficult to destroy with incidental modification but is hard to preserve in cases of malicious manipulation. But, the image authentication scheme is verified by having the sender store the digital signature.

Kundur and Hatzinakos [5] designed a wavelet-based quantization process that is sensitive to modification. The disadvantages are that their method cannot resist incidental modifications and the tampering detection results are very unstable.

Zou *et al.* [9] embedded a verification tag into the spatial domain of the image after having extracted it using a DSA (digital signature algorithm) on the DCT (discrete cosine transform) block. However, their image authentication system can only tolerate JPEG-quality factors greater than or equal to 80, and the algorithm requires extensive computation.

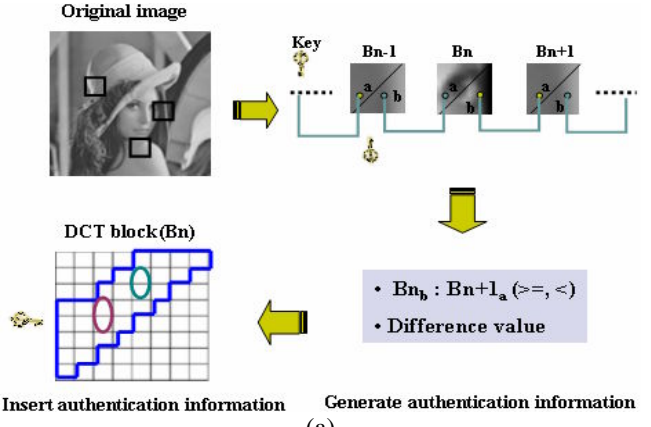
This paper proposes an effective fragile watermarking scheme for image authentication that can detect malicious manipulations while remaining robust towards JPEG lossy compression. An image is divided into blocks in the spatial domain, each block is divided into two parts by randomly selecting pixels, and average gray values for the parts are calculated. The average value is compared with that of the adjoining block to obtain an authentication signature. The extracted information becomes the fragile watermark to be inserted into the image's frequency domain DCT block.

The advantage of the proposed image authentication scheme is that it can easily extract authentication information robust to JPEG lossy compression from the spatial domain and insert it into the host image through a very simple method.

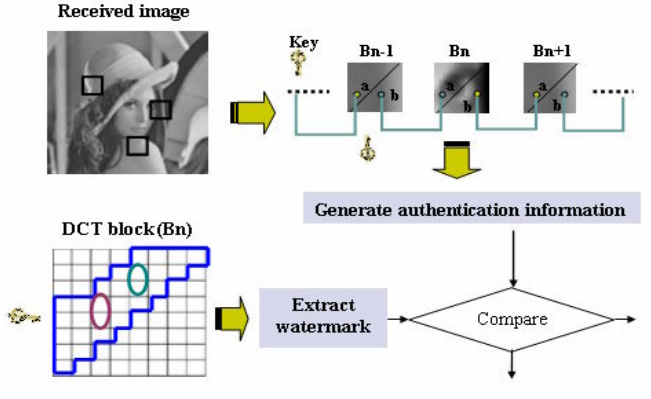
The remainder of this paper is organized as follows. The proposed fragile watermarking scheme is explained in Section 2. Sub-sections of Section 2 describe the generation of the watermark and procedures for its insertion and verification. Experimental results and conclusions are given in Sections 3 and 4, respectively.

2 Proposed Fragile Watermarking Scheme

The proposed fragile watermarking scheme is shown in Fig. 1. The scheme is divided into two parts: (a) generation of authentication signature of the image and (b) subsequent verification of the authentication information of the suspected image with the extracted authentication signature. The two parts are discussed briefly below.



(a)



(b)

Fig. 1. (a) Generating and embedding the watermark, (b) Verification scheme

2.1 Generation of Watermark and Insertion Procedure

First, an authentication signature is extracted from the image's spatial domain as follows. The image is divided into non-overlapping *8* x *8* blocks, which are permuted in the random order generated by the pseudo-random number generator (PRNG) using the seed (key_1) that only authorized users know. Next, pixels are selected randomly from each block; *64* pixels use the seed (key_2). We divide them into *2* parts (Bn_a, Bn_b), with *32* pixels each. An average gray value for the *32* pixels in each of the two parts is obtained. The averages are compared with that of the adjoining block. In other words, of the average pixel values of the two parts, one (Bn_a) is compared with the average pixel value of the one area ($Bn\text{-}1_b$) of the block located in front of the block from the randomly permuted *8* x *8* block, and the average pixel value of the other (Bn_b) is compared with the average pixel value of the one area ($Bn+1_a$) of the block after it.

Average pixel values of the two blocks are compared using signs of (in) equality (>=, <), and the difference between the two pixel values can range from *0* to *255*. Here, we have categorized the differential values into 4 levels. The results of the comparison (>=, <) and the differential is used to create a watermark (authentication signature). In order to maintain the authentication signature obtained from the image during the image compression processing, we modify the gray values of each area within a range that does not damage the original image. If the difference between the average pixel values of the two blocks falls around the boundary values categorized into 4 levels, we modify the pixel values so that the difference is no longer in the range of the boundary value. By adjusting the gray values of each block according to an appropriate threshold, visibility of the image is maintained while enabling extraction of the identical authentication signature for later image verification.

The authentication information generated with this procedure is a fragile watermark, which is inserted into the frequency domain as follows. The *8* x *8* block image is translated into DCT. Next, we insert the magnitude relation acquired in the spatial domain between parts of the adjoining blocks and the differential of the average pixel values into the middle band of the DCT block (*Bn*). In other words, coefficients from 5 groups, *4* from each middle band of the DCT block according to seed (Key_3), are randomly selected. At this time, the 4 groups are modified according to the 4 levels of average pixel value. Rather than the comparison value with the adjoining block, the magnitude information of the average pixel value of the two parts of each single block is inserted for the remaining group. According to the differential value of the average pixels for the 4 levels, one group is selected among 4 DCT coefficient groups. From the selected group, two out of four coefficients are chosen, and the sum of the coefficients is obtained. If the magnitude relation of the sum of the coefficients corresponds with the magnitude relation of the average pixel values of the block areas, the coefficient values are not modified. Otherwise, each coefficient value is modified so that the magnitude relations coincide. At this time, the quantity of the modified coefficients should not change the authentication information acquired in the spatial domain, and maintain the inserted information for image compression processing. Considering visibility of the image, the authentication information is inserted into the middle band of the DCT block.

2.2 Verification Procedure

The verification method for an image is similar to the generation and insertion procedures of the authentication signature. From the spatial domain of a suspect image, an authentication signature is obtained using the identical seed (key_1) used during the generation of watermark. Next, each block is translated to the DCT, and the magnitude relationships are obtained between the sums of coefficients by locating the DCT coefficients divided into 5 groups according to the average pixel differential values of the adjoining block. The magnitude relationship between blocks acquired from the image's spatial domain and the magnitude relationship of the coefficient sums obtained from the DCT block are compared to determine whether the block has been manipulated. A block (*Bn*) is determined to have been manipulated if adjoining blocks on both sides are compared and any one of them is different. For example, if the magnitude relationship, (Bn_b, $Bn+1_a$), of the average pixel values between the rear (right)

block extracted from the spatial domain and the magnitude relationship of the coefficient sums obtained from the block's DCT domain are different, or the magnitude relationship, (Bn_a, $Bn\text{-}1_b$), of the average pixel value between the front (left) block and the magnitude relationship of the coefficient sums obtained from the front (left) block (*Bn-1*) DCT domain are different, the block, (*Bn*), is determined to have been manipulated. However, blocks in the following cases are not considered to have been manipulated. From the verification scheme, if the blocks located at both ends are modified from the randomly permuted block order, the middle block will be designated as manipulated. In order to resolve this problem, the middle block is extracted from what is determined to be *3* consecutive manipulated blocks from the randomly permuted blocks. After returning the block to the original position prior to the random permutation, if the right/left/top/bottom blocks are not manipulated, the block is considered innocent. In an image authentication system, detection results according to malicious manipulations generally focus on a particular area, whereas incidental manipulations, including compression (i.e., filtering and/or sharpening), are indicated throughout the image [11]. Furthermore, the magnitude relationship of the average pixel values in the *2* parts within a single block is compared with the corresponding magnitude relationship of the DCT coefficient sums. If they do not match each other, the block is ultimately determined to have been manipulated.

The proposed image authentication scheme maintains security by using pixel locations divided into the adjoining block and 2 parts within a single block, as well as the seed (*key*) that allows only the authorized users to know the location of coefficients within the DCT block. Moreover, the authentication signature is generated in the image's spatial domain, and the watermark inserted into the image's frequency domain is fragile to malicious modification but robust to JPEG lossy compression. In addition, the watermarked image maintains such visibility that it cannot be distinguished from the original image upon visual comparison. Below, performance analysis is given, and experimental results are presented.

3 Experimental Results

We tested our fragile watermarking scheme with over 100 images. The size of the images used in the experiment was *512* x *512*. The gray value was modified to *5* to maintain the average pixel differential value among each partial block. In other words, if the differential value with the adjoining partial blocks is near the boundary value that is divided into 4 levels, 5 is added to or subtracted from each pixel value. The difference of the sum of coefficient within the DCT block is adjusted to be around 55 as the absolute value of coefficient.

Referring to an experiment on various compression ratios of several images, Table 1 shows the number of error blocks detected for each image's total number of blocks (4096 blocks). In the experiment, images had either less than two or no detected error blocks in JPEG-quality factor 60. Based on the experimental results, we have concluded that our fragile watermarking scheme can be practically applied in image authentication applications.

Table 1. Number of error blocks against JPEG lossy compression

JPEG Compression (QF)	Image					
	Lena	**Barbara**	**Baboon**	**Bridge**	**Goldhill**	**Girl**
90%	0	0	0	0	0	0
80%	0	0	0	0	0	0
70%	0	0	0	0	0	0
60%	0	1	0	0	0	2
50%	36	82	25	46	8	155

Fig. 2 shows the detection results for a manipulated image. The original image is shown in Fig. 2(a), and Fig. 2(b) shows the watermarked image, for which the PSNR is 37.8dB. Fig. 2(c) shows the JPEG-compressed image (QF = 60%), and Fig. 2(d) shows the manipulated watermarked image. In Fig. 2(d), the replaced part of the image is the flower attached to the Lena's hat. Fig. 2(e) shows the detection result when

Fig. 2. Detection result of a maliciously manipulated image. (a) is the original image, (b) is the watermarked image (PSNR = 37.8dB), (c) is the JPEG-compressed image (QF = 60%), (d) is the manipulated watermarked image, (e) is the detection result of the manipulated image only, and (f) is the detection result of the JPEG-compressed image (QF = 60%) after manipulation.

the attack is object placement only, and Fig. 2(f) is the detection result of the JPEG-compressed image (QF = 60%) after being manipulated. As can be seen from the experiment results, tampered regions are sufficiently identifiable although the tampered shape might not be indicated in detail.

4 Conclusions

In this paper, a new image authentication scheme has been proposed, which involves simply extracting an image feature that is robust to JPEG compression from the spatial domain of the image and then inserting it into the frequency domain. The watermark, which is extracted from the image and embedded into the DCT block, is resistant to JPEG compression but is fragile to malicious manipulation.

Future work is needed to ensure that the embedded watermark remains robust at low JPEG compression quality factors and that detection of tampering is indicated more specifically.

References

1. M.Schneider, S.F.Chang: A robust content based digital signature for image authentication. In Proc. IEEE ICIP (1996) 227-230
2. C.Y.Lin, S.F.Chang: A robust image authentication method surviving JPEG lossy compression. In Proc. SPIE Storage and Retrieval of Image/Video Database, San Jose (1998)
3. S.Bhattacharjee, M.Kutter: Compression tolerant image authentication. In Proc. IEEE Int. Conf. on Image Processing (1998) 435-439
4. C.S.Lu, H.M.Liao: Structural Digital Signature for Image Authentication: An Incidental Distortion Resistant Scheme. Proc. ACM Multimedia and Security Workshop at the 8th ACM Int. Conf. on Multimedia, Los Angeles, California, USA (2000) 115-118
5. D.Kundur, D.Hatzinakos: Digital watermarking for telltale tamper proofing and authentication. In Proc. IEEE ICIP (1997) 1167-1180
6. M.Yeung, F.Mintzer: An invisible watermarking technique for image verification. In Proc. IEEE Int. Conf. on Image Processing (1997) 680-683
7. M.Wu, B.Liu: Watermarking for image authentication. In Proc. IEEE Int. Conf. on Image Processing (1998) 437-441
8. P.W.Wong: A public key watermark for image verification and authentication. In Proc. IEEE Int. Conf. on Image Processing (1998) 455-459
9. D.Zou, C.W.Wu, G.Xuan, Y.Q.Shi: A content-based image authentication system with lossless data hiding. In Proc. ICME Int. Conf. on Multimedia and Expo (2003) 213-216
10. C.Y.Lin, S.F.Chang: A robust image authentication method distinguishing JPEG compression from malicious manipulation. IEEE Trans. on Circuits and Sys. of Video Tech. (2001) 153-168
11. C.S.Lu: Multimedia Security. IGP, Hershey London (2005)

PUML and PGML: Device-Independent UI and Logic Markup Languages on Small and Mobile Appliances

Tzu-Han Kao, Yung-Yu Chen, Tsung-Han Tsai, Hung-Jen Chou, Wei-Hsuan Lin, and Shyan-Ming Yuan

Department of Computer and Information Science, National Chiao Tung University, 1001 Ta Hsueh Rd., Hsinchu 300, Taiwan
{gis89539, gis93533, is91027, is91085, is91087, smyuan}@cis.nctu.edu.tw

Abstract. To accomplish developing mobile web applications on variety of mobile execution environments, we propose Pervasive User interface Markup Language (PUML) describing user interfaces for applications on the small devices, and Pervasive loGic Markup Language (PGML) representing the computational logic of the applications. Furthermore, we exploit the XSLT/XPath transformation mechanism to transform documents of PUML/PGML into the target languages, and evaluate them by implementing toolkit and runtime service applications. In Brief, our paper contributes an XML-based application model to assist mobile application developers.

1 Introduction

Currently, there are several mobile execution environments on plentiful mobile appliances. On these environments, XHTML MP (Mobile Profile), Compact HTML (CHTML), etc can be used to develop the mobile web-based applications. To accelerate the development on mobile platform, more some mediator languages are designed, such as XUL [9], UIML [8], AUIML [10], and XIML [11]. XUL (XML User-interface Language) is designed as a cross-platform language to describe the graphical user interfaces of the applications on desktop computers. It works to make the user interfaces of the applications portable.

In [12], we discussed the use of combination of XUL and LGML. In describing user interfaces, we abstracted useful elements from the original XUL elements to describe the widgets on the screens of the display-limited devices. On the logic side, we applied LGML, which is the XML-based language we design to represent the logic. Pervasive User Interface Markup Language (PUML) is the new markup language we have designed to describe user interfaces for applications on the small devices. Moreover, we have revised and improved LGML, and rename it as Pervasive loGic Markup Language (PGML) to state our objective definitely. We exploit the XSLT to transform them into the languages on the mobile environments.

This paper is organized as follows. Section 2 and Section 3, we describe the design of PUML and PGML, respectively. Section 4 presents the transformation mechanism and the simulating result of generating codes. Finally, Section 5 shows the applications; Section 6 discusses the conclusion and the future work.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 947 – 956, 2005.
© IFIP International Federation for Information Processing 2005

2 Pervasive User-Interface Markup Language (PUML)

User interfaces of applications can be divided into two classes roughly. One is plentiful category; the other is fundamental category. The former has abundant widgets to display UI controls. For example, displaying HTML documents on the desktop of PCs has a variety of modules, involving Frames module, Applet module, and etc. The later merely consists of basic presenting modules, such as Form module, Image module. User interfaces of applications on the small and mobile device belong to this kind. A PUML documents is designed to comprises several containers. Each of the containers contains the basic widgets including label, text filed, single-choice listing, multi-choice listing, picture, and action. The following figure shows the conceptual view of PUML. The principles of designing PUML including: (1) user interface abstracting; (2) intermediate language fashioning, and (3) OO (Object-oriented) conceptualizing.

In PUML in the `<puml:user-interface>` element has three types of child elements, which are `<puml:logic-objects>`, `<puml:board>`, and `<puml:layout>`. The `<puml:logic-objects>` element is used to declare the logic object, e.g. .pgml file, and the name used in a PUML document. A `<puml:board>` element can contain `<puml:label>`, `<puml:textnote>`, `<puml:listpaper>`, `<puml:picture>`, and `<puml:action>`, which stand for label, text field, list, image, and action, respectively. These elements are the widgets which are involved in a `<puml:board>` container. Listing 1 illustrates the document structure of a user interface description using PUML.

Listing 1. An overview of the document structure of PUML

```
1   <?xml version="1.0"?>
2
3   <puml:user-interface … >
5    <puml:logic-objects> … </puml:logic-objects>
7    <puml:board … >
8     <puml:logic-objects> … </puml:logic-objects>
10     <puml:textnote … />
12     <puml:picture … />
14     <puml:label … />
16     <puml:listpaper … />
18     <puml:action …> … </puml:action>
19   </puml:board>
21  </puml:user-interface>
```

A `<puml:picture>` element, in a `<puml:board> … </puml:board>` block, can be used to show an image. It has four attributes: `name`, `source`, `altText`, and `align` attributes. A `<puml:label>` is similar to the label of the windows on the PCs. It can display a read-only text string on the screen. Its main attribute is `show Text`, which can be assigned a string value to be displayed on the screen. We can use this element to show some title or prompt information. Oppositely, `<puml:textnote>` can get a string from the user's input. The `<puml:listpaper>` element serves as a choice group for picking a single item of a listing, or

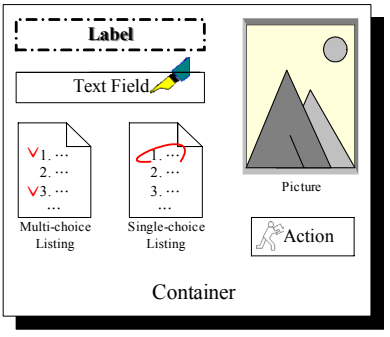


Fig. 1. The conceptual view of PUML

selecting a group of options among the items of the listing. The are two elements(<puml:action> and <puml:buttom>) designed for event-binding in PUML v1.4. <puml:action> elements can be rendered to the submission items of a form. While users press a submission button of a form, the request parameters inputed in the form are sent to the server side by `get` method in HTTP protocol. <puml:buttom> is used to show the button in the interaction UIs. This element has three attributes including `name`, `showText`, and `action`. The two formers provide the same meaning mentioned above. Users can specify the address in the attribute action field.

3 Pervasive loGic Markup Language (PGML)

In PUML, the design principles of the language we consider are: (1) computation generalizing, (2) intermediate language fashioning, and (3) OO conceptualizing. The first means that, logic languages, such as C, Java, or WML Script, have the primary statements to declare variables and function blocks, and flow-control and condition-control mechanisms to complete basic computation. Therefore, we abstract the primary expressions and statements to define PGML. From the three points, we make PGML own the following capabilities: the object-oriented concept, variables declaration, flow-control, condition-control statements, and method declarations. Peripherally the features of PGML and PUML are different, but essentially the two

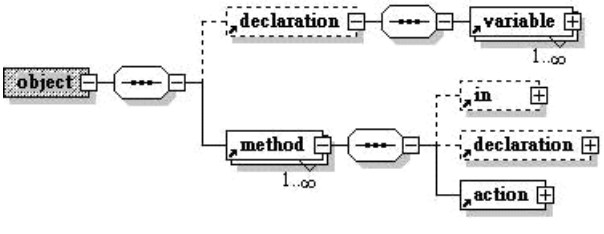


Fig. 2. A tree view of the XML Schema of PGML

languages have the same intention. Figure 2 demonstrates a PGML object in a tree view of the XML Schema [13]. An object, e.g. a PGML file, is composed of a variable declaration <declaration> and a least one method <method>.

Programmers can use PGML elements to write the computational logic. For instance, the <pgml:method> element expresses a method declaration block. The <pgml:in> element contains some child elements, which are the arguments needed passing into this method. In PGML, there is a element <pgml:init> similar to the <pgml:in> element. Differently, it is used to declare and initiate the local variables in a flow-control element <pgml:for>…</pgml:for>, or a method declaration block <pgml:method>…</pgml:method>. The following listing demonstrates a PGML example, and Listing 3 shows the Java source program transformed from the PGML code.

Listing 2. A method declaration in PGML

```
1 <?xml version="1.0"?>
2 <pgml:object name="addTwoNum" version="1.0"
3  xmlns:pgml="http://dcsw3.cis.nctu.edu.tw/Project/
4  pervasive/PGML/" >
5
6 <pgml:method name="sum"  visibility="public" returntype="int">
8      <pgml:in>
9        <pgml:variable name="num1" type="int" />
10       <pgml:variable name="num2" type="int" />
11     </pgml:in>
13     <pgml:action>
14       <pgml:return>
15         <pgml:add>
16           <pgml:operand select="num1" />
17           <pgml:operand select="num2" />
18         </pgml:add>
19       </pgml:return>
20     </pgml:action>
22</pgml:method>
24</pgml:object>
```

Listing 3. The Java code transformed from the PGML document in Listing 4

```
1   package SumExample;
2   import SumExample.*;
3   import java.lang.Integer;
4   import java.lang.String;
5
6   public class addTwoNum {
8     public addTwoNum(){ }
9     public int sum( String  sys_num1, String sys_num2 ){
10      int  num1= Integer.parseInt(   sys_num1);
11      int  num2= Integer.parseInt(   sys_num2);
12      return ( num1 + num2 );
13    }
14 }
```

4 Leveraging XSLT/XPath Transformation

4.1 An Overview of Language Transformation Mechanism

In the implementation, we use an XSLT engine to transform parsed PUML trees into the result trees. Next, stream results will be serialized into the source programs of the target languages (e.g. WML, ASPX, etc). The process contains the following four main steps:

PUML Well-Formedness Checking and Validating: When finishing writing a PUML document, a programmer can use the development toolkit to check the well-formedness and validity for the PUML/PGML document.

PUML Parsing: With no error found in the document, the document will be parsed into a DOM tree.

PUML Transforming: The PUML/PGML transformer, deriving from the transformation engine Xalan-Java 2.5.1 [12], is capable of transforming the tree into the result trees through each transformation stylesheet for the target languages. Each of stream results (`javax.xml.transform.stream.StreamResult`) a result tree is generated. Besides, the stream results are serialized into source programs. The transformation procedure will be explained in the next section.

PUML Compiling: MExE Language Compiler (mexe-compiler) can compile each of the source programs generated into its specific executable code. For example, for the J2ME and PJava platforms, a Java source program (a .java file) generated is compiled into Java bytecodes (.class files). Nonetheless, not all the generated source programs need to be compiled. .wml (WML) and .wmls (WML Script) codes need no compilation, since they can be interpreted by user agents of client devices. From this point, if the source programs of Java were generated, they would be compiled into executable codes by the mexe-compiler. Otherwise, when the programs of WML and WML Script are generated, they are not compiled.

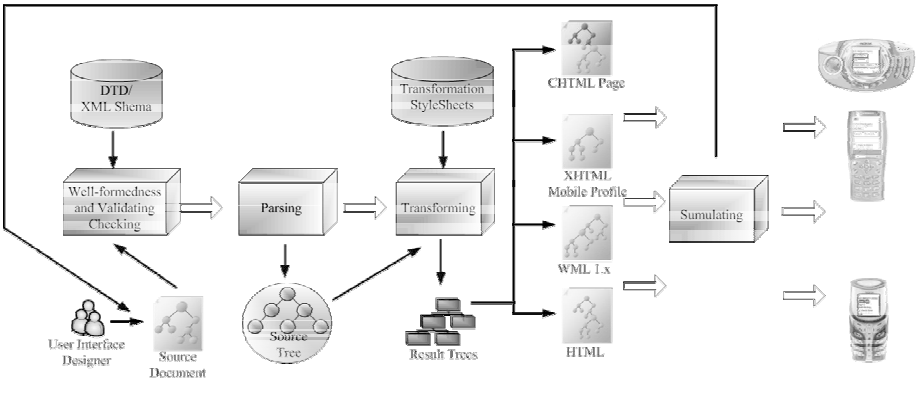


Fig. 3. the well-form checking, parsing, transforming, and simulating sequence

Table 1. The mappings from PUML tags into WML expressions in the PUML-to-WML transformation stylesheet

	PUML Tag	MIDP Expression
Root element	<user-interface>	javax.microedition.lcdui.MIDlet
Container	<board>	javax.microedition.lcdui.Form
Text String	<label>	javax.microedition.lcdui.StringItem
Text Field	<textnote>	javax.microedition.lcdui.TextField
Selection List	<listapper> <item> … </listpaper>	javax.microedition.lcdui.ChoiceGrop
Image Display	<picture>	javax.microedition.lcdui.ImageItem
Action Trigger	<action>	javax.microedition.lcdui.Command

5 Applications

5.1 An Example

In the section, we will demonstrate an example written in PUML and PGML documents, and the simulating results of the generated J2ME and WML codes from applying the transformation mechanism, mentioned above. Listing 4 is a PUML document (UIExample.puml). There are three <puml:textnote> elements in the listing. The first two can get two input numbers, and the last can display the result by summing the two numbers up. The <puml:action> … </puml:action> block describes that the two inputted values are passed into the sum method of object1, e.g. addTwoNum.pgml (declared in Line 12-14). Furthermore, the value attribute of the <textnote> widget, sum in board2, will be updated by the retuned value after the action is triggered. Line 27 describes the code to accomplish that. The section of addTwoNum.pgml can sum the two inputted number.

The following code (Listing 5) is the WML code which generated from UIExample.puml. In the code, there are three <input> elements which are converted from the three <textnote> elements in Listing 6. The top three of Figure 4 demonstrate simulating the WML code generated. For example, a user inputs two number, 1 and 2, in Step 1, and selects action 2 in Step 2 subsequently. Then, the <go> would be performed. However, there is a problem—how to accomplish passing the values of the numbers into the WMLS function, and updating the value attribute of the <input name="sum" … /> element—must be coped with. The trick we used is exploiting WMLBrowser.getVal() and WMLBrowser.setVal(). WMLBrowser.getVal() can be used to get the value of the variable specified from the WMLBrowser environment. Specifically, the values of the variables, standing for the two <input>, can be obtained by invoking WMLBrowser.getVal() in the WML Script. Returning the computed result can be completed through

`WMLBrowser.setVal()`. addTwoNumBroker.wmls is the code generated in transformation to manipulate this event-handling. The related usage of the two methods can be referred in [14].

Listing 4. A user interface described in PUML

```
<?xml version="1.0"?>
<puml:user-interface name="UIExample" version="1.2"
 xmlns:puml="http://dcsw3.cis.nctu.edu.tw/Project/
 pervasive/PUML/">

  <puml:board name="board1" title="FirstPage" >
    …
  </puml:board>

  <puml:board name="board2" title="SecondPage">

    <puml:logic-objects>
      <puml:object name="object1"  source="addTwoNum.pgml" />
    </puml:logic-objects>

    <puml:label name="mainTitle"
     showText="Input two numbers:" />
    <puml:label name="num1Title" showText="Number 1:" />
    <puml:textnote name="num1" value="0" />

    <puml:label name="num2Title" showText="Number 2:" />
    <puml:textnote name="num2" value="0" />

    <puml:label name="sumTitle" showText="Sum:" />
    <puml:textnote name="sum" value="0" />

    <puml:action name="action2" showText="action2">
      <puml:change container="board2"
       component="sum" update="value">
        <puml:use-object name="object1" method="sum">
          <puml:param select="num1" />
          <puml:param select="num2" />
        </puml:use-object>
      </puml:change>
    </puml:action>
  </puml:board>
</puml:user-interdace>
```

Fig. 4. The WML code generated from the code of PUML/PGML

Listing 5. The WML code transformed from the PUML code in Listing 6

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
  "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>

  <card title="FirstPage" id="board1">
    …
  </card>

  <card title="SecondPage" id="board2">
    <p>Input two numbers:</p>
    <p>Number 1:</p>
    <p><input name="num1" value="0"/></p>
    <p>Number 2:</p>
    <p><input name="num2" value="0"/></p>
    <p>Sum:</p>
    <p><input name="sum" value="0"/></p>
    <do name="action2" type="accept" label="action2">
      <go href="addTwoNumBroker.
       wmls#start('board2action2')"/></do>
  </card>

</wml>
```

5.2 A Web Service by Using XML Web Services

The following (Figure 5) demonstrates the architecture which provides the transformation mechanism as the web services. This architecture has two advantages: (1) diverse front-end side and extensibility, and (2) version control. The former is that we

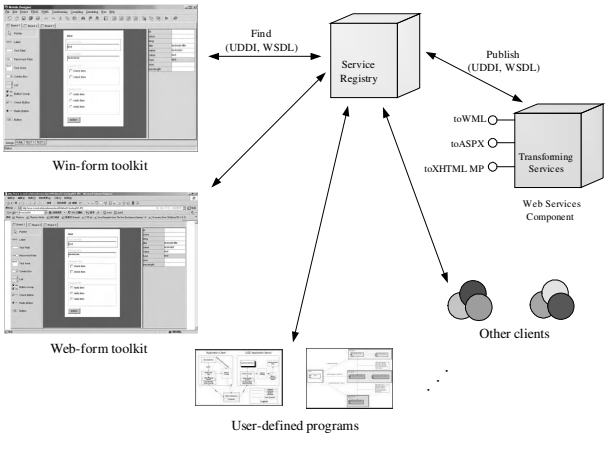


Fig. 5. The overview of enveloping the transformation function by using XML web services

can design not only win-based toolkit, but also construct a web-based toolkit for programmers to design UIs through the Internet. The program of the front-end side is not restricted. You can develop your applications and then use the interfaces of the transformation functions to generate WML, CHTML, and XHTML MP pages by using XML Web Services invocation. The latter indicates that the original components can be replaced easily with new ones.

Listing 6. An overview of the document structure of PUML

```
Server side
[WebMethod]
public XmlDocument  TransformToLang(String lang, XmlDocument  inPUML){
   XmlDocument resultNode = transformers.Transform(lang, inPUML);
   return resultNode;
}
Client side
...
WindowsApplication1.TransForm.AdaptationServer tfcl=new WindowsApplica-
tion1.TransForm.AdaptationServer();
tfcl.Url=(string)HMtrans[name];
((TextBox)page.Controls[0]).Text=tfcl.TransformTest(page.Text, myXmlDo-
cument);
...
```

6 Conclusions and Future Work

In this paper, we introduced the design of PUML/PGML and the applications for the small and mobile devices. Programming by using XML shows the following advantages: (1) transforming a PUML source into others in the target languages easily, and (2) neglecting which target environment to run applications when writing the programs of the applications. They can create form interaction block of the web pages by using PUML. However, it is not user-friendly for programmers to write PUML document directly. To improve that, we are planning to design a toolkit module which enables programmers to construct PUML from existing HTML pages in drag & drop way. Besides, toward the pervasiveness, we will explore the mechanisms of transforming PUML not only into CHTML [4], XHTML [5], JavaSwing [6], but also into .NET Mobile Pages [1], XForm [7], and the representation of the small IA devices.

References

1. Microsoft Mobile Web Forms. http://samples.gotdotnet.com/mobilequickstart/(mgk4rd2jnyo1zm55tgnot02p)/Default.aspx.
2. Ricardo Devis, The Object-Oriented Page. June 1997. http://www.well. com/user/ ritchie/oo. html
3. Xalan-Java 2.5.1. http://xml.apache.org/xalan-j/.
4. Tomihisa Kamada. Compact HTML for Small Information Appliances. Feb 1998. http://www.w3.org/TR/1998/NOTE-compactHTML-19980209.
5. Steven Pemberton, Daniel Austin, Jonny Axelsson, et al., XHTML(TM) 1.0 The Extensible HyperText Markup Language (Second Edition). W3C Recommendation. 1 August 2002. http://www.w3.org/TR/xhtml1.

6. David M. Geary. Graphic Java 2: mastering the JFC 3rd ed. Sun Microsystems. 1999.
7. Micah Dubinko, Leigh L. Klotz, Jr., Roland Merrick, et al. XForms 1.0. W3C Recommendation. October 2003. http://www.w3.org/TR/xforms/.
8. Marc Abrams, Constantinos Phanouriou, Alan L. Batongbacal, Stephen M. Williams, Jonathan E. Shuster. UIML: An Appliance-Independent XML User Interface Language. http://www8.org/w8-papers/5b-hypertext-media/uiml/uiml.html.
9. Neil Deakin. XUL Tutorial , July 2002. http://www.xulplanet.com/tutorials/xultu/.
10. Roland A. Merrick and et. al. AUIML: An XML Vocabulary for Describing User Interfaces. May 2001. http://www.belchi.be/download/merrick.pdf.
11. Angel Puerta and Jacob Eisenstein. XIML: A Common Representation for Interaction Data. Proceedings of the Sixth Intelligent User Interfaces Conference (IUI 2002). San Francisco, California, USA. January 2002.
12. Tzu-Han Kao, Sheng-Po Shen, Shyan-Ming Yuan, and Po-Wen Cheng. An XML-based Context-Aware Transformation Framework for Mobile Execution Environments. Lecture Notes in Computer Science (LNCS) of Springer-Verlag (APWeb 2003, Xian, China), Vol. 2642 / 2003, pp. 132 - 143.
13. David C. Fallside. XML Schema Part 0: Primer. W3C Recommendation. May 2001. http://www.w3.org/TR/xmlschema-0/.
14. WAP Forum. WMLScript Specification Version 25-Oct-2000, Sept 2001. http://www.wapforum.org/.

Distributed Contextual Information Storage Using Content-Centric Hash Tables

Ignacio Nieto, Juan A. Botía, Pedro M. Ruiz, and Antonio F. Gómez-Skarmeta

Departamento de Ingeniería de la Información y las Comunicaciones,
Universidad de Murcia, Spain*

Abstract. We analyze the problem of storing contextual information across a set of distributed heterogeneous mobile devices. These devices form a Mobile Ad Hoc Network (MANET) interconnected to Internet, in which partitions may eventually happen due to mobility. We present a new mechanism called Contextual Hash Table (CHT), which uses the semantic of the contextual information to select which nodes will store that information. Unlike previous general-purpose data replication algoritms, CHT manages to store information in the area of the network in which that information is most likely needed by neighboring nodes.

1 Introduction

Future mobile applications and services are expected to adapt and interact with their environment, providing a richer set of services to end users. This kind of applications are usually known as context-aware applications, or even ubiquitous applications. These applications use information about their environment (known as *contextual information*) such us user preferences, location, device capabilities, etc. Based on this information, they can provide a wide range of services like adapting contents to the capabilities of the user's device, providing information about nearby objects, etc.

The management, adquisition, representation and storage of contextual information, defined as any information about a user and its sorrounding, becomes a central aspect of any ubiquitous computing platform. For the representation and adquisition of context, we already developed an Open Context Platform (OCP) which uses an ontology represented using Ontology Web Language (OWL). In this paper, we focus on how to store an access that contextual information in a distributed network consisting of mobile heterogeneous devices.

Mobile Ad Hoc Networks (MANETs), consist of a number of mobile devices equipped with a wireless interface, making them very flexible and easy to deploy. These networks are distributed in nature, and do not rely on any centralized entity. Nodes are free to move, and they use neighboring nodes as relays, to send data to nodes which are not within their radio range. All these properties, make

* This work has been partially funded by the ENCUENTRO (00511/PI/04) research project of the Seneca Foundation and the CICYT research project FUZZY-KIM (TIC-2002-04021-C02-02) of the Spanish Government.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 957–966, 2005.
© IFIP International Federation for Information Processing 2005

ad hoc networks an ideal candidate to support ubiquitous computing environments. MANETs can also connect to fixed networks like the Internet, creating a so-called hybrid MANET. Howerver, the network may be partitioned and merged at any time due to mobility. Thus, every service within the MANET must work in a distributed way whether there is Internet connectivity or not.

Throughout the rest of the paper, we assume an ubiquitous computing scenario consisting of a set of users and its corresponding devices. Devices are heterogeneous and may have different computing capabilities. In addition, devices will form an hybrid MANET, providing connectivity among devices. As in any MANET, the network may get temporarily partitioned, preventing the communications among some set of nodes. Devices will be responsible of obtaining contextual information, and store it within the network (eventually at different nodes). In addition, a device will be able to locate and access contextual information stored in any other node. In this paper we introduce CHT, a novel approach based on hash tables toefficiently deal with contextual information in such an scenario.

2 Related Work

Our concern here is related to information dissemination however, context information is not needed to be known by everyone in the system. Instead, this kind of information has to be available on demand (i.e. context information refers to dynamical situtation of users and neither a node needs to know the context fo all users nor all nodes need to know the context of a single user). Hence, here we are not referring to dissemination to all nodes but to a few of them, appropiately selected. Moreover, the kind of network we have is a mobile ad hot network (MANET). Both of these facts (i.e. ad hoc and device mobility) introduce additional complexity to the problem. However, there are examples in the scientific literature which study gossiping [1] and multicast [5] in MANET. In our context, data dissemination consists on storaging and accessing data by name. This data tokens will be located in a concrete node (and perhaps in a number of additional ones). These nodel will usually be mobile, hence location of data tokens depend on location of hosting nodes. One interesting approach using geographical information for dissemination of information is the GLS (Grid Location Service) [10]. The GLS system uses GPS information for location and information dissemination. The approach we present in this article uses indoor location information, based on the Wifi network of the building [14]. Recent work which store and access data tokens by name can be found in [12,13]. Storing and accessing data by name implies using a mechanism based on a hash table [11]. The scenario that the authors are focused on consists on a sensor network and they use a distributed has table inside a technique called *Data Centric Storage* or *DCS*. In DCS, each node is target of some data identified by means of a hash function. We will call this node the *home node* of the data. Whenever an event is generated, the sensor that detected it (we can think of this sensor as the "source" of the event) sends this event to its *home node*, or the nearest node in case the *home*

node is not available. If a concrete contextual information needs to be accessed, the hash function has to be applied to obtain its *home node*. Both operations can be done in $O(\sqrt{n})$.

We may also find several studies about data distribution and management in Ad Hoc networks. Data caching in an Ad Hoc environment has been treated in [3], where data transmitted in the network and the routes used to transmit it are both cached in the path of the information. In [4], an integrated data lookup and replication scheme is presented where a predictive mechanism is used to predict likely group partitioning of nodes. ADS [2] is a middleware service for Ad Hoc environments to share and receive information from the neighbourhood applied to CARLA, a M-Learning scenario. This approach relies on *information markets*, well-known places where different kinds of information from multiple applications can be pooled and exchanged. Regarding to data replication, in [6], a peer-to-peer service model for managing, searching, and streaming partitioned media objects is described, paying special attention to replication strategies for media segments. Data replication has also been treated in works such as [8] as the right approach for managing data avaliability in Ad Hoc networks, where the topology gets modified unpredictably mainly due to the mobile nature of the nodes.

3 Problem Statement

We are interested in ubiquitous computing scenarios in which users, user different heterogeneous devices, and communicate among them wirelessly without the need of any centralized authority. This scenario, which is the most typical case, is characterized by its intrinsic mobility. Users will use a MANET providing connectivity among them, and eventually to applications and services over the Internet. Although Internet connectivity cannot be assumed to be always present. Our interest is in designing efficent contextual information distribution algorithms for these kind of scenarios.

3.1 Assumptions and Requirements

The particular problem of information distribution in ubiquitous computing scenarios based on MANETs has some distinguishing features from traditional scenarios. Hence, we work under the following assumptions. (1) We assume that user devices are heterogeneous in terms of capabilities and computing capacity. They can be laptops, PDAs, mobile phones, etc. (2) The environment is dynamic. Devices can show up and leave the system at any time, even without prior notice (e.g. due to network disconections, battery exhaustion, etc.). (3) Users are mobile. They can freely move throughout the ubuiquitous scenario. This places an important requirement onto our system, which should be able to deal with that mobility of the users and the eventual partitions of the underlying network. (4) The underlying ad hoc network has limited resources. The proposed solution shall try to minimize the consumption of network resources by placing

the information in appropriate locations. (5) Every node in the system has an unique identifier.

Within the rest of this section we describe design alternatives for our solution.

3.2 Design Alternatives

As we stated before, our goal is to efficiently distribute contextual information. That is, information about entities, objects and locations of the environmentin which the users are moving around. Every device in the system will have an unique identifier. Contextual information will be generated in a concrete node, and the system must assign a *home node* where to store it.

There is a wide range of alternatives for distributing information in such an ad hoc scenario with our requirements. One extreme is storing every contextual information in the node that generated it, taking order O(1) in terms of network messages. In this case, queries from the rest of nodes to obtain this information will require a broadcast in the entire network (we don't know in which node the data is a priori). So the cost of accessing the information requires O(n), where n is the number of nodes in the network. That is, n messages are needed for the broadcast plus $\sqrt{n}$ for sending the answer back to the querier, being $\sqrt{n}$ the mean path length.

Another extreme in the range of solutions is storing all the contextual information generated by any of the nodes replicated in every node in the network. In that case, performing the storage of each data requires a broadcast, which has a cost of O(n) in terms of the number of messages. In this case, queries can be resolved locally in order O(1).

There are a number of other intermediate solutions based on the idea of replicating the contextual data only in a set of nodes, but reaching a good tradeoff is not an easy task. In addition, almost in all these cases, the mechanisms require a broadcast operation, which tends to be costly for these environments, even if broadcast reduction techniques like common dominating sets (CDS) are used. Thus, we target a different approach based on hash tables and a hierarchical structure to achieve a better solution. In addition, by distributing the date based on their content, we can better select *home nodes* to reduce the cost in most common operation scenario.

As we mentioned, we will use a different approach based on a hash function to distribute the information across the nodes. Figure 1 shows a typical layout of a set of devices in the MANET, grouped by their location. In our example, we assume we are inside a building, and nodes are distributed all over the building. Nodes form an Ad Hoc network that may or may not have connectivity with external networks like the Internet. If there exists such connectivity, then we may have also one SCPP taking control of a group of areas (e.g. nodes in certain rooms). We will denote by n_i the diameter (expressed in number of nodes in the longest path) of the area i. We also denote by N the diameter of the whole area controlled by the SCPP. Every time we need to know the *home node* of a given contextual information (in case we need to update this information or we need to access it) we will call a hash function returning the *home node* of this

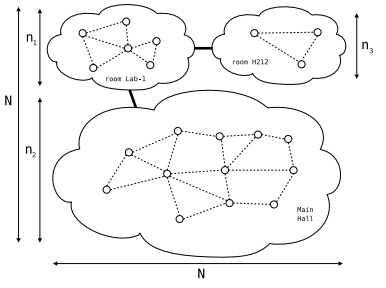


Fig. 1. Typical layout of our environment using CHT

information. This operation can be performed in order O(1). If the *home node* of that information lies in the same room or area we can obtain the information with two messages (answer and response). Which means in $O(\sqrt{n_i})$ transmissions of each intermediate node because $\sqrt{n_i}$ is the mean path length for nodes within the area n_i. In the worse case in which the route to the *home node* is not kwnon, a limited broadcast can be used, having a cost of $O(n_i)$ messages. Our hash function will be designed so that contextual information will be placed close to the area in which there are more nodes interested in. Thus, the access cost will be reduced. In the unlikely case in which a node would be interested in contextual information of other areas, the access time would be $O(\sqrt{N})$ in the average case and $O(N)$ in the worst case,in which we don't know a route to the destination and we need to flood the whole area.

4 CHT: Contextual Hash Table

This section describes in detail the CHT algorithm for distributing contextual information represented according to our Ontology Context Platform (OCP). In section 4.1 we describe the algorithm in detail, showing concrete examples. In adition, in section 4.2, we discuss the issue of replication and we introduce some replication mechanisms for CHT to guarantee the avaliability of data it handles even when mobility and node failure rates increase.

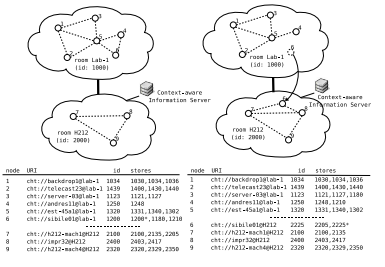
4.1 The CHT Algorithm

As we stated before, each node producing, consuming or storing contextual information in the system will have one unique identifier. Provided that our OCP works with OWL-represented onlogies, in which identiers are URIs, our identifiers will have the following format:

```
URI = cht://resourceDescription@physicalLocation
```

where *resourceDescription* is the description name of the node (a.k.a. resource) that must be unique in the system. Additionally, *physicalLocation* is the name of

the physical area in which the node is currently located. In case a node does not know its physical location, it will use "unknownLocation" as default. Some examples of CHT URIs are `cht://backdrop1@room1` and `cht://printer2@labH44`. Besides its URI, each node in the system will generate a unique identifier expressed as a long integer. This identifier will be temporal, and will be regenerated if the user changes from one location to another (i.e: the user changes to another room in the building). Given that users are most interested in the contextual information related to their current location, our CHT will mostly consider the location as the most important piece of contextual data, when selecting *home nodes*. Thus, contextual information is expected to be stored in the same area in which the user is. In fact, location is the most important contextual information in mobile context-aware scenarios[9,7]. Hence, the location of the user will be the dominating factor when selecting its temporal identifier. So, the higher order bits of the identifier will be taken from the location of the node (e.g. identifier of the room obtained from the ontology). The rest of the identifier will be formed from the context of the user or node, its URI and/or the arriving order of the nodes to the room. This identifier (a long integer, such as 25000) will be used as a reference to discriminate the contextual information that must be stored at the user or node. That's to say, given the identifier returned from the hash function, the *home node* for that contextual information will be the one whose hash value equals or is closer to that identifier than any other node. According to our requirements, the identifiers of two nodes that are in the same area will always be closer than the identifiers of two nodes in different areas.



node	URI	id	stores
1	cht://backdrop1@lab-1	1034	1030,1034,1036
2	cht://telecast2@lab-1	1439	1400,1430,1440
3	cht://server-03@lab-1	1123	1121,1127
4	cht://endres1@lab-1	1250	1240
5	cht://est-45a1@lab-1	1320	1331,1340,1302
6	cht://sibile01@lab-1	1200	1200*,1180,1210
7	cht://h212-mach1@H212	2100	2100,2135,2205
8	cht://impr32@H212	2400	2403,2417
9	cht://h212-mach4@H212	2320	2320,2329,2350

node	URI	id	stores
1	cht://backdrop1@lab-1	1034	1030,1034,1036
2	cht://telecast2@lab-1	1439	1400,1430,1440
3	cht://server-03@lab-1	1123	1121,1127,1180
4	cht://endres1@lab-1	1250	1240,1210
5	cht://est-45a1@lab-1	1320	1331,1340,1302
6	cht://sibile01@H212	2225	2205,2225*
7	cht://h212-mach1@H212	2100	2100,2135
8	cht://impr32@H212	2400	2403,2417
9	cht://h212-mach4@H212	2320	2320,2329,2350

Fig. 2. CHT operation

Invoking the hash function with a cocrete contextual information will give us a hash value dependent, as said before, on a concrete contextual information and a source URI. We will use this value to identify the *home node* of that information by means of a simple check of integer proximity, computing the minimum:

$$\min_{1 \leq i \leq n} abs(id - id_i)$$

where id_i is the identifier obtained from node i, id is the hash value obtained for the contextual information and N the total number of nodes. Using an ordered distributed identifiers table, this could be done in small time.

Figure 2 shows a concrete example of the CHT algorithm operation. Our example consits of two interconnected rooms: "Lab-1" and "room H212". Each room has several nodes we have labelled from 1 to 9. Each node has a unique URI and an identifier it generated when it entered the room. Each node stores contextual information we have identified by its hash value. Both identifiers and hash values are represented by long integers. This way, node i stores the information with hash value v if there is no other node j with a nearer identifier to v. That is to say:

$$home - node(context) = i \Longleftrightarrow \nexists j | abs(j - v) < abs(i - v)$$

Notice that the six nodes of the room "Lab-1" have their identifier closer to 1000 (the identifier value associated to that location) because the location is the most important factor when assigning node identifiers. Similarly, the three nodes of the room "room H212" have their identifiers close to 2000. Down in the same figure we can see the hash values of the information they currently store. In the situation depicted on the upper part in the figure, node 6 leaves room "Lab-1" to enter room "room H212". Before leaving the room, node 6 announces their neighbours of its intentions, and transmits to them the contextual information that must stay in the room. The node has at that moment three bits of information: local information concerning its own preferences and the device characteristics, data that has no more relevance for the nodes of the room "Lab-1", and two contexts of interest for the nodes of the room (identified with hash values 1180 and 1210). These two bits of information must be kept in the room, so node 6 distributes them to their nearby nodes, nodes 3 and 4 respectively.

Upon entering the room "room H212", node 6 modifies its URI from `sibile01@lab-1` to `sibile01@H212`, and its identifier also gets modified from 1200 to 2225. Its contextual information, now adapted to show the change in location (besides any other contextual change occurred in the process), has now been labeled with hash value 2225. The node then performs a broadcast to any other node in the new room announcing its presence. The nodes then add its URI and identifier to their tables and, checking their own contextual informations, node 7 notices the contextual information with hash value 2205 it was storing is now nearer to the new node, so it transmits this information to node 6.

With CHT integrated with OCP, the contextual information is relocated as the users move throughout the building. By definition, neighboring nodes will also have close identifiers. When a user leaves a room and enters another (see Fig.2), it passes the room-dependent information (contextual information about objets, people or devices in the room) to other nodes in the room, dropping this information if is not of its interest now. When the node enters the new room it receives a new dynamic identifier, and its new neighbours determine wich information should the new node store. The node is now the new "nearby" node to some other nodes and the *home node* of some contextual information. Similarly, the contextual information of the room has changed with the entrance of the new node, and this newly generated contextual information will be distributed and replicated over the nodes of the network as exposed previously. With CHT, the

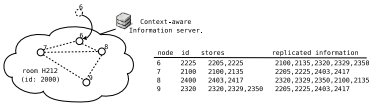


Fig. 3. Data replication in CHT

contextual information moves through the building continuously as a distributed informational entity.

4.2 Replication of Contextual Information

To enhance the basic operation described so far, we use a replication mechanism to guarantee that contextual information is not lost due to node failures or network partitions. The CHT replication mechanism consists of maintaining copies of the contextual information of every node in their upper and lower nearby nodes (in terms of their identifiers). As an example, consider the figure 3, where node 9 will store apart from its own contextual information, that of nodes 6 and 8. This information will be stored as "replicated information", and kept apart from the contextual information that has node 9 as *home node*. The "border nodes" (i.e: nodes that have a minimum or maximum hash value so are missing a upper or lower nearby node, like node 8) will replicate their information on their only nearby node and the other only node with only one neighbour (the node with maximum identifier will store the contextual information from the node with minimum identifier and viceversa). As an example, figure 3 shows that node 8 has no upper nearby node, so it will replicate its contextual information on nodes 6 (its lower nearby node) and 7 (the node with minimum identifier). This mechanism works remarkably well and shows great scalability as number of nodes in the system increases.

The replication schema must be taken into account specially when a node enters or leaves the system. In the example shown in figure 3, when node 6 enters room H212, it stores its own contextual information now identified with hash value 2225, receives the contextual information identified with hash value 2205 from node 7 as *home node* contextual information, and stores the contextual information of nodes 7 and 9 as "replicated information". Notice that node 6 will only receive the target information of nodes 7 and 9 (specifically contextual informations identified with values 2100, 2135, 2320 , 2329 and 2350), not their replicated information (like the replicated information node 9 stores from node 8, its upper nearby node). Similarly, when a node exits a room, the nodes must drop or redistribute both their target contextual informations and their replicated contextual information properly.

With this mechanism, if a node fails or leaves unexpectedly, its contextual information is not lost, because their nearby nodes already possess that information, and they only must set that replicated information as target information if they are now the *home nodes* of that information. They will detect it because the previous *home node* is no longer reachable. As an example, we assume that

node 9 in figure 3 dies unexpectedly due to an energy failure. When the node failure is detected, the remaining nodes inmediatly adjust their contextual identifier tables to reflect the missing node, and nodes 7 and 8 (the nearby nodes of node 9) re-evaluate the replicated information they are storing from node 9 (i.e: contextual information labeled with hash values 2320, 2329 and 2350). All this contextual informations are now nearer to node 8 than 7, so node 8 happens to be the new *home node* of them, so it sets them as target information, dropping them from their replicated information area. Node 8 also sends this three contextual informations to node 6, its lower nearby node, that node 6 stores as replicated information.

Now we will calculate the chance of loosing a concrete contextual information with our replication method. We will call N the number of active nodes in some area of the system, and we may assume a probability of p of a node failure without loose of generallity. In order for a contextual information to get lost all three nodes carrying that information (its *home node* and their two nearby nodes) must fail simultaneously. The probability that this will happen is $(\frac{p}{N})^3$. As an example, with a N value of 100 nodes and a failure probability of 5% we have a probability of $1,25E^{-10}$ of a contextual information loose. This schema guarantees with a high probability that our contextual information will be always available to users, an important factor if we want to handle a trusted distributed contextual storage and management system.

5 Conclusions and Future Work

In this paper we present a contextual information distribution algorithm that distributes the information according its semantic content. CHT (Contextual Hash Table) approachs the problem of information distribution with a hash table strategy, where the information is distributed according to its contextual significance. A replication mechanism guarantees that the information will always be available to other users, and the hash table approach offers a balanced storage of the information and a reduced access time, both for storage and recovery of the contextual data.

References

1. N. Bauer, M. Colagrosso, and T. Camp. An efficient approach to distributed information dissemination in mobile ad hoc networks. Technical Report Technical Report MCS-04-01,, The Colorado School of Mines, February 2004.
2. Matthias Brust, Daniel Görgen, Christian Hutter, and Steffen Rothkugel. Ads as information management service in an m-learning environment. *Knowledge-Based Intelligent Information & Engineering Systems*, 2004.
3. Guohong Cao, Liangzhong Yin, and Chita R. Das. Cooperative cache-based data access in ad hoc networks. *Computer, vol. 37, no. 2*, pages 32–39, February 2004.
4. Kai Chen and Klara Nahrstedt. An integrated data lookup and replication scheme in mobile ad hoc network. 2004.

5. M. Hauspie, A. Panier, and David Simplot-Ryl. Localized probabilistic and dominating set based algorithm for efficient information dissemination in ad hoc networks. In *Proceedings of the 1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS'04)*, Fort Lauderdale, Florida, USA, 2004.
6. Shudong Jin. Replication of partitioned media streams in wireless ad hoc networks. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 396 – 399, New York, NY, USA, October 2004.
7. F. Bennet D. Clarke J.B. Evans A. Hopper A. Jones and D. Leask. Piconet: Embedded mobile networking. *IEEE Personal Communications, 4(5)*, pages 42–47, 1997.
8. Goutham Karumanchi, Srinivasan Muralidharan, and Ravi Prakash. Information dissemination in partitionable mobile ad hoc networks. In *Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems*, page 4, Washington, DC, USA, October 1999.
9. R. H. Katz. Adaptation and mobility in wireless information systems. *IEEE Personal Communications, 1(1)*, pages 6–17, 1994.
10. Jinyang Li, John Jannotti, Douglas S. J. De Couto, David R. Karger, and Robert Morris. A scalable location service for geographic ad hoc routing. In *Proceedings of the 6th ACM International Conference on Mobile Computing and Networking (MobiCom '00)*, pages 120–130, Boston, Massachusetts, August 2000.
11. W. W. Peterson. Addressing for random access storage. *IBM Journal of Research and Development*, 1(2):130–145, 1957.
12. Sylvia Ratnasamy, Deborah Estrin, Ramesh Govidan, Brad Karp, Scott Shenker, Li Yin, and Fang Yu. Data-centric storage in sensornets. *SIGCOMM 2002, February 1st*, 2002.
13. Sylvia Ratnasamy, Brad Karp, Scott Shenker, Deborah Estrin, Ramesh Govindan, Li Yin, and Fang Yu. Data-centric storage in sensornets with ght, a geographic hash table. *Mobile Networks and Applications*, 8:427–442, 2003.
14. Ignacio Nieto Carvajal Juan A. Botía Blaya Pedro M. Ruiz Antonio F. Gómez Skarmeta. Implementation and evaluation of a location-aware wireless multi-agent system. *Proceedings of the International Conference on Embedded and Ubiquitous Computing*, 2004.

An Integrated Scheme for Address Assignment and Service Location in Pervasive Environments

Mijeom Kim[1], Mohan Kumar[1], and Behrooz Shirazi[2]

[1] The University of Texas of Arlington,
Box 19015, Arlington, TX 76019, USA
{mkim, kumar}@cse.uta.edu
[2] School of EECS, Washington State University,
Pullman, WA 99164, USA
shirazi@eecs.wsu.edu

Abstract. We propose an efficient scheme called CoReS (Configuration and Registration Scheme) that integrates address assignment and service location for ad hoc networks prevalent in pervasive computing environments. CoReS exploits node heterogeneity such that more capable and stable nodes serve others. CoReS allocates addresses to individual nodes locally, but employs global allocation states to handle network merge situations. In addition, CoReS exploits the positive features of distributed directory services to perform service location in a centralized manner resulting in minimal communication overheads. We analyze the characteristics of CoReS architecture, evaluate its performance and compare with other schemes. Through the evaluation and comparison, we demonstrate that the integrated CoReS system exhibits high efficiency and cross-layer optimization.

1 Introduction

Pervasive computing is an emerging and very challenging area of research, envisaging numerous computing devices within a small area [1]. Devices possessing specific computing and sensing capabilities may want to communicate with each other to maximize users' quality of life. Such devices may need to form multi-hop mobile networks without prior infrastructures and make use of services available on other nodes. Efficient address assignment and service location are necessary for autonomic network formation and continuous collaboration among devices.

Routing protocols assume that every node in the network has a unique address and is aware of the address of the destination node it is communicating with. Perhaps, the greatest underlying question in networking for pervasive computing does not lie just in providing seamless mobility, but in supporting zero-configuration networking capabilities [2]. Some interesting proposals have come up in the area of automatic address configuration [3][4][5][6][7][8][9]. In addition, service discovery in pervasive computing environments especially for ad hoc networks has started to receive attention [10][11]. Conventionally routing protocols and service discovery protocols are placed in different layers, but integrated or tightly coupled versions of these two pro-

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 967 – 976, 2005.
© IFIP International Federation for Information Processing 2005

tocols have the ability to enhance efficiency especially in mobile ad hoc networks where each node needs to perform routing and service discovery [12].

In this paper, we propose CoReS (Configuration and Registration Scheme), an integrated protocol for address assignment and service location in pervasive computing environments. CoReS is a distributed, yet hierarchical approach using a small subset of the nodes called CR-nodes that are responsible for assigning addresses to other nodes and registering their services. Relatively stable and capable nodes serve as CR-nodes, thus CoReS exploits node heterogeneity in terms of mobility and capability. A node that has registered its services with a CR-node is referred as a *child node* of the CR-node. The CR-node that has the registration entry of a child node is referred as the *default CR-node* for that child node. CoReS is not limited to any particular MAC protocol or physical network media. Without loss of generality, in this paper, we assume multi-hop wireless ad hoc networks that are common in pervasive environments.

CoReS is adaptable to both IP and flat address formats. For small networks such as Personal Area Networks (PANs) or sensor networks, which do not need any Internet protocols or applications, flat address format is preferable because smaller address size can be used. CoReS assigns unique addresses serially using the *mod* operation, starting from a random reference address. With CoReS, new address allocation is performed locally and network merge situation is managed with global address allocation state. In addition, service location of CoReS operates in a centralized manner, yet flexibly through cooperation of distributed directory services.

The rest of the paper is organized as follows: in Section 2, we discuss related work on address auto-configuration and service discovery. Basic mechanism and protocols of CoReS are presented in Section 3. Section 4 presents protocols handling challenging situation such as network merge. In Section 5, we analyze characteristics of CoReS architecture and evaluate the protocols by comparisons with other schemes. We conclude and discuss future work as possible extensions to CoReS in Section 6.

2 Related Work

Address auto-configuration has been a topic of research in various types of networks. Generally, there are two types of auto-configuration protocols - stateless and stateful. Protocols utilizing a stateless approach [3][6] do not maintain allocation states, thus new joining nodes self-configure themselves with randomly selected or h/w based addresses. To assure the uniqueness of the chosen address, Duplicate Address Detection (DAD) is needed causing high overhead/latency and scalability problems due to flooding. In network merge situation, stateless approaches require each node responsible for duplicate addresses since the allocation state is not maintained anywhere.

In stateful approaches, address conflicts are avoided as allocation is managed. Dynamic Host Configuration Protocol (DHCP) is one of the most popular protocols. DHCP maintains allocation states at a central server, thus not applicable in pervasive environments where accessibility to the server is not guaranteed always. More recently, there are interesting proposals employing a stateful approach in Mobile Ad hoc Networks (MANET). One initial effort is MANETConf [4] where every node maintains global allocation states. A new address allocation requires consensus of all nodes, incurring high communication overhead/delay as well as high memory usage at

each node. In network merge situations, detection of address conflicts is easy since global allocation information is maintained at each node. However, nodes with duplicate addresses need to tackle the conflicts themselves.

Another popular stateful approach exploits a buddy system [5][8] where disjointed allocation information is maintained locally at every node. Whenever a new node joins, a preconfigured node just splits its address pool and assigns the divided pool to the new node, ensuring unique address allocation. Zhou et al. proposed a unique address allocation scheme, which uses a special function to assign an address to each node for large scale MANETs [7]. These schemes [5][7][8] enjoy low communication overheads for address allocation procedure without network wide flooding. However, they have limitations in such dynamic situations as network merging and/or abrupt node departure since there is no global view of allocation status.

Service discovery architectures/protocols have been developed and investigated with an emphasis on auto-configurable networks. Sun's Jini [13], Service Location Protocol (SLP) of IETF and Microsoft's Universal Plug and Play (UPnP) are prominent among them. Jini operates with centralized directory servers and UPnP performs in a decentralized manner. SLP works in the two operation modes with and without centralized directory services. All the above protocols are not well suited for pervasive environments due to limitations in supporting node mobility or scalability.

There have been several proposals for service discovery in ad hoc networks recently [10][11][12]. Schemes in [10] and [11] achieve flexible service matching by employing a service hierarchical tree to describe and discover services. However, all nodes are required to maintain the fixed hierarchical tree and employ complex matching mechanisms. In [12], authors proposed the service discovery architecture that is supported by the network layer. Besides the formation of a virtual backbone, the periodic service registration and management of multicast group incur high overhead.

3 CoReS Architecture/Protocols

Each node maintains its degree of mobility (d_m) and resource value (R). d_m is a relative value to indicate dynamism of the node and a function of average speed of the node. For static nodes, $d_m = 0$. R is a quantitative value to indicate the amount of resources the node has. There are three types of resources that make up R - computation, communication and energy. Using d_m and R, each node sets utility value ($\delta = f(d_m, R)$) that is proportional to R and inversely proportional to d_m. The nodes with high δ are good candidates for CR-nodes.

A CR-node predetermines its workload by setting address range (size of address pool) for its child nodes. The address pool size is set to be proportional to utility value (δ) of the CR-node. A CR-node assigns a unique address to each joining node serially ensuring no address duplication unless there are more nodes than total global addresses. After getting an address, a new node registers its service functionalities and its δ with its default CR-node. A CR-node maintains a child table and a CR table. A child table maintains local address allocation states with service information and the utility values of child nodes for directory service. A CR table maintains global address states with information of all CR-nodes: i) addresses of CR-nodes; and ii) address ranges that CR-nodes manage.

A CR-node creates another CR-node among its child nodes when it has exhausted a predetermined number of addresses, a *threshold* of the given address pool. A child with the highest δ is selected for a new CR-node. A new CR-node gets the CR table from its default CR-node and sets its address range starting from the next available address.

3.1 Address Auto-configuration and Service Registration

When a node N_i wants to join a network, it sends one-hop broadcast join message to its neighbors. If no reply is received before the given *timeout*, N_i tries again with longer *timeout* until the number of retrials exceeds its predefined value ω, or it receives a reply from a neighbor. ω is set to be inversely proportional to δ. After ω number of retrials without receiving any reply, N_i assumes that there is no CR-node. Thus N_i becomes the first CR-node and sets own address pool range starting from a randomly chosen address within the global address space. After that, the first CR-node assigns itself the first address of the own address range and self-registers its service in the child table. In addition, N_i announces its creation to the whole network to let other (possibly) existing nodes register with N_i. When several nodes try to start the network simultaneously, the node with the lowest ω value becomes the first CR-node because it has relatively higher resources and lower mobility.

Let us consider a general situation where some CR-nodes exist. N_i's neighboring node, say N_j upon receiving the join message, sends a probe message to its default CR-node. The default CR-node in turn replies to N_j with remaining address count (A_c). N_j forwards the reply that includes A_c to N_i. Suppose N_j is a CR-node, it directly replies to N_i with its A_c. After the given period of time, N_i selects the neighbor node N_k as a *broker*, where N_k is the neighbor responding with the highest A_c. N_k's default CR-node assigns an address for N_i through the broker N_k as shown in Fig. 1.

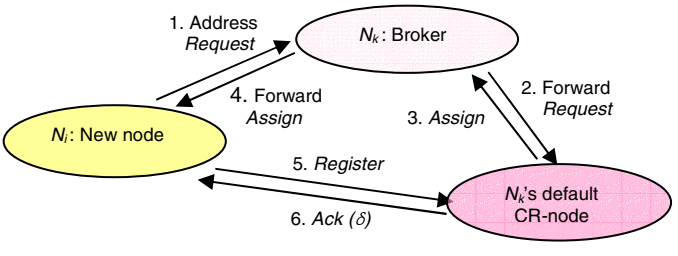


Fig. 1. Node Join Operation: After getting an address, N_i registers its service with the default CR-node that in turn sends an acknowledgement with its δ

3.2 Service Location

When a node needs to avail of a service, it sends a service request message to its default CR-node with the service description. After receiving the service request message, the default CR-node looks up its child table to find out matched services. Additionally, it multicasts the service request message to other CR-nodes. Any CR-node

that has the information about the requested service in the child table forwards the request message to the child node that provides the requested service.

We call a node requesting a service, *a client* and a node providing the service, *a server*. When a server receives a forwarded service request from its default CR-node, it needs to decide whether to accept the service request according to its current conditions such as workload and available resources. The server may either reject the service request by simply ignoring it, or accept it by replying to the original client along with the detailed service specifications including current resource status and cost. The client selects a server that provides the best preferable service by comparing several replies with details of service specifications.

Late service de-registration may be performed at the CR-node as follows. When a server receives a forwarded service request message, the server needs to send acknowledgement to its default CR-node. If the CR-node does not get acknowledgement within the given number of trials, it suspects that the server (the child) has left and marks the server in the child table. For the marking, one bit of flag field is used in child tables. Similarly, CR tables also have the marking field. When a CR-node has used up all addresses in its pool, it tries to contact the marked child nodes again. If the CR-node still cannot hear from the marked child nodes, it de-registers the child nodes' services and caches their addresses for next address allocation. However, before de-registration of a child, whenever the CR-node realizes the child node is present in the network, the CR-node resets the marking field. We call this *lazy service de-registration* because de-registration does not take place when the server leaves, but is delayed until there is no available address at the default CR-node. The lazy service de-registration makes the protocol more flexible by handling the situation where nodes temporarily drift from the network and come back later.

4 Robustness of CoReS

4.1 Loss of CR-Nodes

CR-nodes are not expected to leave often because they are relatively stable and have more resources including energy. When a CR-node *A* needs to leave, it is responsible to create a substitute CR-node. If *A* has a capable child node (that has utility value higher than or equal to that of *A*), *A* makes the child a substitute. Otherwise *A* asks a peer CR-node to make a substitute. However, nodes moving out of network or nodes that fail due to some unexpected reason do not have any prior knowledge that they are separating from the network. As a result, a CR-node may leave abruptly without handing over to a substitute, resulting in loss of the CR-node. This case is handled starting from the marking process similar to that in lazy service de-registration. Whenever a node sends a message to a CR-node, the former is supposed to get acknowledgement from the latter. A node suspects loss of a CR-node when the former fails to get acknowledgement from the latter within a given number of retrials with increasing *timeout*. A node suspecting loss of a CR-node, say *A*, floods this information to the network. (If several nodes suspect the loss simultaneously, the situation is handled with *content flooding* explained in Subsection 4.3.) When other CR-nodes acquire the information about the loss, they set the marking field of *A* in their CR tables.

When a child node of *A* gets this information, the following algorithm is executed at the child (δ_A is a utility value of *A*):

```
If (need to increase service utilization) {                1
  If (δ ≥ δ_A)                                             2
    become a substitute CR-node                            3
  else {                                                   4
    relinquish the current address                         5
    get a new address from another CR-node                 6
    reregister services with the new default CR-node       7
  }                                                        8
}                                                          9
else {                                                    10
  wait until a default CR-node is detected again          11
  if (the detected CR-node is a substitute one)           12
    reregister service with the new default CR-node       13
}                                                         14
```

During the missing period of the default CR-node, a child node sends service requests to another CR-node through neighbors.

After marking, if a CR-node *B* wants to create a substitute CR-node, first, *B* needs to wait for a predefined time and check again for presence of the marked CR-node *A*. If *B* still cannot hear from *A*, the former confirms the latter has left without informing, and creates a substitute among its child nodes by assigning the same address range of *A*. When a child of *A* becomes a substitute CR-node (line *3* in the above algorithm), the child gets the CR table information from another CR-node through neighbors. The substitution process of the child also includes the waiting and checking procedures. The substitute CR-node announces the substitution by flooding. After receiving the announcement, the remaining child nodes of *A* reregister with the substitute CR-node (line *13* in the above algorithm) and the other CR-nodes update their CR tables.

Before the substitution, if the marked CR-node appears again, other CR-nodes reset the marking field in their CR tables and the appearing CR-node synchronizes its CR table with the current one. Notice that substitution of a lost CR-node is not required urgently. By the waiting and checking procedures before substitution, temporary CR-node drift can be handled.

4.2 Creation of a CR-Node

Creation of a new or substitute CR-node requires consensus among existing CR-nodes in the system. When a CR-node *A* wants to create another CR-node, first it needs to check a possible ongoing creation event. If *A* has a pending creation event saved in its cache, it waits for completion of the event before requesting for permission. Otherwise, *A* directly requests other CR-nodes for permission to create. Another CR-node, say *B,* upon receipt of the request, sends a negative reply if it has a previous pending creation event. Otherwise, *B* designates the creation event of *A* as pending and sends a positive reply.

If *A* receives positive replies from all other CR-nodes other than marked ones, it creates a CR-node. If *A* receives any negative reply, it needs to wait until completion of the ongoing creation event and try again by asking for permission. When *A* does not get a reply from some CR-nodes, *A* starts the marking process by repeating the

permission request (explained in Subsection 4.1). After completing the marking process, *A* resumes creation of the CR node. After a new (or substitute) CR-node is created, it informs to all other CR-nodes of its creation to let them update their CR tables and remove the pending creation event.

4.3 Networks Merge and Partition

When several networks merge, the addresses of all nodes in smaller networks will change by adding offset to larger networks. Let us consider a scenario when three networks *X*, *Y* and *Z* merge. Table 1 shows their address ranges and required changes.

Table 1. Merging of three networks

Network	Original address range (address pool size)	Needed change	Address range after merge
X	67 ~ 159 (93)	+ 0	67 ~ 159
Y	110 ~ 166 (57)	+ 50 (159 – 110 +1)	160 ~ 216
Z	25 ~ 43 (19)	+ 192 (216 -25 +1)	217 ~ 235

Suppose $size(X) \geq size(Y) \geq size(Z)$, where $size(N)$ is the size of address pool assigned in network *N*. *X* does not need to change any address in its network. All addresses in *Y* need to be changed by adding 50 (159 - 110 + 1) to the original ones. Similarly, network *Z* starts its address from 217 by adding 192 (216 – 25 + 1).

CoReS needs network ID to distinguish nodes belonging to different networks [4]. When a CR-node *A* conceives network merge situation, it floods a merge message including its CR table and network ID to the combined network. When another CR-node *B* receives the merge message from *A* for the first time, it compares *A*'s network ID with its own. If they are same, *B* just retransmits the merge message. Otherwise, *B* saves *A*'s CR table along with network ID and retransmits the merge message. In addition, *B* floods another merge message including own CR table as well as network ID. After a fixed time period, each CR-node in the system collects all CR tables corresponding to different networks. When there are overlapped address pools among the CR tables, CR-nodes calculate the needed address changes for each network and merge the CR tables. In addition, CR-nodes let the other nodes in the system know the needed address change by flooding. All flooding in this subsection operates somewhat different from the general flooding protocol such that the messages with the same payload are treated as duplicates regardless of the sources of the messages. In other words, regardless of which and how many nodes initiate the flooding, all nodes send the message once to their neighbors by ignoring any duplicate message of the same content. We refer to this flooding protocol as *content flooding*.

When no address ranges of the merging networks are overlapped, address reconfiguration is delayed until a situation (like another merge) causing overlapped address pools occurs for the case of temporary network merger. When one part of the network separates from the rest, reconfiguration is needed in both partitions. The partition that has less than half CR-nodes, changes its network ID. Regardless of getting the new

network ID, each network partition needs to handle two side effects as follows: i) missing of some child nodes - use the lazy service de-registration in Subsection 3.2; and ii) loss of CR-nodes - apply the protocol for loss of CR-nodes in Subsection 4.1.

5 Evaluation

5.1 Analysis and Comparison of Auto-configuration Schemes

Auto-configuration protocols using a stateful approach maintain address allocation information in several ways: i) centralized global allocation states; ii) distributed global allocation states; and iii) distributed local allocation states. Central global allocation states are unsuitable for pervasive environments due to uncertain availability of the central server. Distributed global allocation states in [4][9] need to be synchronized among all nodes, causing high overhead/latency. In contrast, distributed local allocation states allow concurrent assignments with low cost as in [5][7][8], but network merge situation cannot be handled efficiently due to lack of global allocation view.

CoReS exploits a stateful approach requiring special nodes (CR-nodes) to maintain distributed allocation states. CR-nodes maintain global allocation states in CR tables as well as local allocation states in child tables. Therefore, address allocation is performed locally and network merge situation is handled with global view of allocation states. In this subsection, we investigate and compare CoReS with other auto-configuration schemes in terms of handling of network merger, address (as resource) management and scalability.

First, CoReS handles network merge situation efficiently due to maintenance of global allocation states. The number of total broadcasts (content flooding) to handle network merger is the number of merging networks + 1 (dissemination of a CR table per network and dissemination of the needed address change). The key advantage is that each node in smaller networks can resume ongoing communications swiftly since the node can apply the same conversion rule to all addresses it maintains including its destination nodes' addresses and cached addresses (routes) in routing tables.

Second, in terms of address resource management, CoReS uses address resource optimally by serial assignment while some schemes [6][7] need significantly larger global address space compared to the number of nodes. Most stateful schemes [4][5][8][9] use address resource efficiently, but they have some difficulty in handling abrupt node departure or temporary node drifts while CoReS handles reclamation of addresses and temporary node drifts in a natural way with the marking process.

Third, with regard to scalability, CoReS is scalable compared to those with distributed global allocation states due to the difference of synchronization requirements: protocols using global distributed allocation states need to synchronize the information among all nodes while CoReS requires only CR-nodes to synchronize their address pools (CR tables) regardless of individual address allocation. Stateful schemes with local allocation states seem efficient for basic address assignment, but are not scalable facing challenging situation such as network merge or abrupt node departure.

5.2 Performance Comparison of Service Discovery Protocols

We compare CoReS with well known MANET broadcast based service discovery protocols that employ pull and push models. Pull scheme operates such that whenever a client needs a service, it floods a service request message and any server providing the matched service replies to the client. Push scheme performs such that whenever a client needs a service, it looks up its own cache for information of the needed service. If the client cannot find a match in its cache, it needs to wait for the service advertisement before utilizing the service. A server floods its service advertisement periodically. When a client receives a service advertisement, the client caches the advertisement if it is interested in using the service in future.

We compare the service discovery protocols in terms of operating cost (c_o) measured by the number of exchanged messages for a client to find out all matched services (matched servers) in the system. Let us say n is the number of total nodes and n_s is the number of matched servers in the system. To make things simple, we assume that there is no message loss, thus the system can discover n_s matched services.

In CoReS, a client sends a service request message to its default CR-node that in turn forwards the request to the other CR-nodes in the system. The number of request messages sent (and forwarded) to all CR-nodes is equal to the number of CR-nodes (n_{cr}). After receiving the request messages, CR-nodes that have information of the requested service forward the message to n_s servers (n_s messages). Finally, n_s servers reply to the client directly (another n_s messages). Thus, with CoReS, $c_o = (n_{cr} + 2 \times n_s)$.

When a node floods a message in the network, every node in the network forwards the message once ignoring duplicates. Hence, we assume that n messages are needed for one flooding for simple comparison. With pull scheme, a client floods a service request and the n_s servers reply to the client, which leads to $c_o = (n + n_s)$.

With push model, c_o is dependent on the probability that a client has cached information of the requested service. The probability depends on cache policy and cache size that is another complex issue. Here, we simply refer to the probability that a client has no cached information for the matched service as P_{miss}. Generally, as network size (in terms of n) increases, P_{miss} increases due to limitation of cache size. With P_{miss}, c_o is given by $(n_s \times P_{miss} \times n)$ since $n_s \times P_{miss}$ advertisements are required.

Usually $n_s << n$ and $n_{cr} << n$, thus service discovery of CoReS incurs a lot less operating costs than pull scheme. In turn, operating cost of pull scheme is less than that of push scheme when $n_s \times P_{miss}$ is high, e.g. greater than 1.

6 Conclusion and Future Work

We propose CoReS, an efficient and integrated protocol to assign addresses for nodes and to support service location in local pervasive environments. CoReS exploits unevenness of nodes to allow more capable and stable nodes to serve others. CoReS employs the hierarchical and distributed architecture such that address assignment is performed locally and network merge situation is handled with global view. Service discovery in CoReS operates efficiently compared to distributed approaches based on broadcast, especially in terms of operation costs.

There are several issues we plan for future work. First, we will utilize CR-nodes for other network functionalities such as routing and security to justify the cost of building and maintaining the CoReS architecture. Second, trust management will be deployed to establish reliable relationship among entities for overall secure interactions. Third, we try to provide locality between child nodes and their default CR-nodes to improve system performance and scalability. Finally, we will incorporate context awareness to CoReS operations for pervasive environments.

Acknowledgements

This material is based on work supported by the National Science Foundation under Grant No. 0129682.

References

1. Saha D and Mukherjee A, "Pervasive Computing: a Paradigm for the 21st Century", IEEE Computer, March 2003, pp 25 – 31
2. P. Mahonen, J. Riihijarvi, M. Petrova, and Z. Shelby, "Hop-by-Hop Toward Future Mobile Broadband IP", IEEE Communications Magazine, Volume 42, Issue 3, March 2004.
3. Zero configuration networking group. http://www.ietf.org/html/charters/zeroconf-charter.html. Cited 29, January 2004
4. S. Nesarig and R. Prakash, "MANETConf: Configuration of Hosts in a Mobile Ad Hoc Network," Proc. IEEE INFOCOM 2002, pp 1059-1068
5. A. P. Tayal and L. M. Patnaik, "An Address Assignment for the Automatic Configuration of Mobile Ad Hoc Networks", Personal and Ubiquitous Computing Volume 8 , Issue 1, 2004
6. C. Perkins, J. Malinen, R. Wakikawa, E. M. Belding-Royer and Y. Sun, "IP Address Autoconfiguration for Ad Hoc Networks", IETF, MANET Working group, November 2001
7. H. Zhou, L.M. Ni and M. W. Mutka, "Prophet Address Allocation for Large Scale MANETs", Proc. IEEE INFOCOM 2003
8. A. Cavalli and J. Orset. "Secure hosts auto-configuration in mobile ad hoc networks", To appear, Ad Hoc Networks Journal by Elsevier Science, 2004
9. M. Mohsin and R. Prakash, "IP Address Assignment in a Mobile Ad Hoc Network," IEEE MILCOM, volume 2, October 2002, pp. 856 – 861
10. D. Chakraborty, et al, "Towards Distributed Service Discovery in Pervasive Computing Environments", IEEE Transactions on Mobile Computing, July. 2004.
11. S. Helal, N. Desai, V. Verma and C. Lee, "Konark--A Service Discovery and Delivery Protocol for Ad-hoc Networks," Proc. the Third IEEE Conference on WCNC, March 2003.
12. Ulas C. Kozat and Leandros Tassiulas, "Network Layer Support for Service Discovery in Mobile Ad Hoc Networks", Proc. IEEE INFOCOM 2003
13. Sun Microsystems, Jini Architecture Specification, June 2003

Modeling User Intention in Pervasive Service Environments

Pascal Bihler, Lionel Brunie, and Vasile-Marian Scuturici

Laboratoire LIRIS – UMR 5205, INSA de Lyon,
7 avenue Jean Capelle, F-69621 Villeurbanne cedex, France
{pascal.bihler, lionel.brunie, marian.scuturici}@insa-lyon.fr

Abstract. The introduction of pervasive computing environments in everyday life will not just be a big step for users, but also for application designers. The well defined interaction interfaces will make place for other, more intuitive ways of interaction. It is the challenge for a pervasive system middleware to capture and model the user intention in a smart way and to solve ambiguousness in the user's expression of a pervasive action. This paper introduces the Pervasive Service Action Query Language (PsaQL), a language to formalize the description of a user intention using composed pervasive services. The work describes a way of translating the user intention into an executable action and propose algorithms performing this translation. Considerations to implement this process are given within the scope of PerSE, a pervasive service environment developed by our research group, together with general evaluation metrics for such algorithms.

1 Introduction

"Pervasive" or "Ubiquitous" Computing – as Mark Weiser calls it in his trendsetting paper "The Computer for the 21st Century" [1] – left its cache and starts to become everyday reality. A network of omnipresent, highly embedded computing machines (possibly with limited power and communication resources) seems reasonable in the upcoming century in a manner that one even does not recognize the presence of these computers anymore. In the way, one perceives (or do not perceives anymore) his/her[1] intelligent environment, the way one interacts with it has to change as well. The shift is clearly defined from teaching the user how to interact with a computer to teach the computer how to interact with a user.

Computing devices will move from reactive actions to proactive ones. Today, most machines react to more or less formal commands given by the user, e. g. switching the television channel if the user presses some button on his remote control at the beginning of the advertising block. In the future, knowing the user's dislike for this kind of information service, a proactive television could immediately switch the channel, mute the screen and sound for the duration

[1] In the following, we will just use the male form, but the female form is intentionally included.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 977–986, 2005.
© IFIP International Federation for Information Processing 2005

of the advertisement or present background information about the interrupted broadcasting from the Internet.

In our understanding, "being proactive" means for a computing system "understanding the user's intention", "learning from its action history", and "proposing an action" or "acting". These two first aspects touch the two ways of deriving the action intention, either from an explicit user directive (voice command or similar) or by comparing the context information with the action history. In a pervasive environment bringing different services on different machines together, it should not be the concern of the service developer to care about the user intention interpretation, but rather he should rely on a well defined service interface and concentrate on doing the service work well.

At the LIRIS laboratory in Lyon, we develop a pervasive service environment platform called PerSE. This paper presents the first step towards interpreting a user intention and describes in a formal way a corresponding action using service composition. The proposed approach is integrated in the PerSE environment.

The rest of this paper is organized as follows: The challenges and constraints of a pervasive service environment are presented by introducing the main aspects of PerSE in Sect. 2. In the following Sect. 3, the process of handling the user intention in a pervasive service environment is presented. We introduce an intuitive formal language which can work as an intermediate step between the user's expression of an action intention and the system internal action representation. The translation process between the user's and the machine's interpretation is presented. This is followed by some benchmarks guidelines in Sect. 5, leading to a short overview of related work, a conclusion, and a set of open questions.

2 PerSE: A Pervasive Service Environment

In this section we present the pervasive service environment PerSE developed by our research group. This allows us to introduce in an informal way the key concepts of this paper that will be formally defined in the next sections.

Pervasive service environments support the interaction of independent services collaborating to perform an intended action. Examples for such services are a filesystem interface, a translating service, a laser pointer acting as an input device or a video projector to visualize information. It is the task of the middleware to connect the services in a pertinent and efficient way. We call this combination of services *complete action*.

PerSE models such a system and offers the interfaces needed to use the managed services without worrying about the limits of the pervasive environment. The basic architecture of PerSE is designed like this: Each *service* is managed by a base-application called *PerseBase*, corresponding to a device included in the pervasive environment. The PerseBase, which is connected to other PerseBases, is responsible for managing the services it proposes and is capable to construct and start complete actions. These complete actions are modeled in PerSE as connected graphs of services.

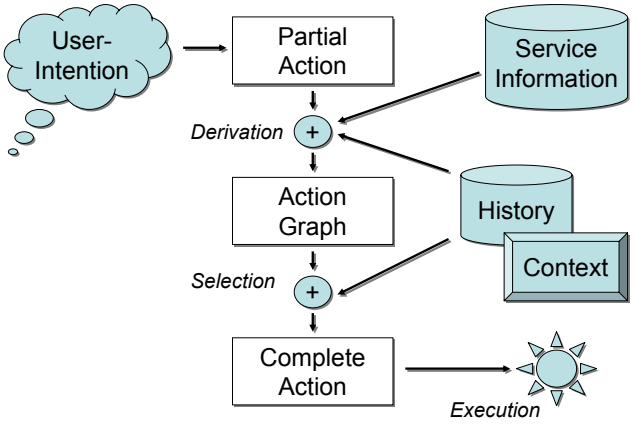


Fig. 1. A user intention is transformed into a complete action by selecting the optimal action from all possible complete actions (action graph) matching the user intention, using the knowledge about available services, the context information and the execution history

The environment must select the best action as answer to a user intention with respect of the constraints of the whole system (i. e. data transmission speed, reliability, . . .). To describe a user intention, we introduce the concept of a *partial action*, that means a description of the action containing (more or less) exactly defined the data source and the data sink and maybe some steps between. The PerseBase in charge has to derive a *complete action* from this partial action and the knowledge it has about the available services in the network. This process is sketched in Fig. 1. The challenge of deriving this complete action from a partial user input is studied in this work. Detailed information about this process can be found in Sect. 3.

The process of transforming a user intention into an executable series of commands for the pervasive environment can easily be understood with the following example, which will accompany us through the rest of this paper:

A person enters a room and wants to display some presentation about his vacation from his personal notebook. With today technologies and softwares, he has to connect his computer to the local network and copy the file containing the presentation to the local server connected to the video projector. Then he has to find the appropriate program on this machine to present his slides. Alternatively, he can deconnect the projector from the local machine and connect it to his own mobile computer, adjust the screen settings and so forth. A PerSE*-enabled, smart classroom, does not provide the ability to deconnect the local projector cable or to access the room server, because these entities are "invisibly" embedded into the room arrangement. Instead, the user expresses his desire by saying or typing "Show the sunrise presentation on the projector". His notebook, which runs a PerseBase as well and communicates with the local resources via Bluetooth or WLAN, interprets the user command as a partial action with a service delivering*

a presentation*, the attribute* sunrise *for this service, and a connection with a service called* projector*. Using the information about the local available services, context information (for instance the room the user has entered) and the action history (maybe the user has already ran a request like this in the past, so if this action is still valid, it can be reused), the PerseBase constructs a graph of all possible (and reasonable) service combinations, each representing a complete action matching the given partial action (see Fig. 1). From these combinations, an algorithm (see Sect. 4) selects the "best" one depending on a given cost-function, e. g. the volume of transferred data. Finally, this complete action is executed: the presentation is displayed on the projector.*

3 Modeling User Intention - PsaQL

Even if the user interface once will be hidden completely into the background, there will still be the need of expressing a partial action in a formal way. This representation can be used directly to express an intention or as an exchange format used by parts of the PerSE middleware system. Therefore we decided to define a formal language, PsaQL (Pervasive Service Action Query Language). PsaQL plays a similar role in the PerSE-enabled pervasive environment as SQL [2] does for accessing relational database management systems. PsaQL also looks similar to SQL on the first glance (BNF notation):

```
<partial_action> ::= USE <action_part> [<ext_action>]
<ext_action> ::= WITH <action_part> [<ext_action>]
<action_part> ::= <attr_constr_def>[ FOR <service_constr_def>]
                                    [ ON <base_constr_def>] |
                  <service_constr_def>[ ON <base_constr_def>] |
                  <base_constr_def>
<base_constr_def> ::= BASE <base_constraint>[ AS <name>]
<base_constraint> ::= <name> | LIKE "<partial_name>"
<service_constr_def> ::= SERVICE <service_constraint>[ AS <name>]
<service_constraint> ::=  <name> | LIKE  "<partial_name>"
<attr_constr_def> ::= (<name> | LIKE "<partial_name>")[ AS <name>]
<name>  ::= {<'a'-'z', 'A'-'Z', '0'-'9','_'>}
<partial_name>  ::= {<'a'-'z', 'A'-'Z', '0'-'9','^','$',
                      '(',')','[',']','.','+','*','?', ...>}
```

Our example (see Sect. 2) can be expressed in PsaQL as:[2]

```
USE sunrise.ppt ON BASE notebook WITH SERVICE projector
```

[2] In this case the user already has the knowledge about the current device names. If the user is not sure about these parameters, his request could be expressed using the LIKE-statement, which currently advises the algorithm in charge to interpret the associated character chain as a regular expression.

4 From User Intention to User Satisfaction

We present in this section an algorithm to translate a partial action into a complete action (see Fig. 1). A *partial action* is a formal representation of a user intention whereas a *complete action* is a connected graph of services, representing the best combination of services to satisfy the user intention. The translation process consists of three steps:

1. Translate the user-input (given for instance in PsaQL) into an internal model of a partial action
2. Expand the partial action to an *action graph* using the service description database, the action history, the context information and heuristic strategies
3. Select the best solution as complete action

With some mathematical formalization, we can define step two and three of this algorithm:[3]

Definition 1 (Partial Action). *Let B be the set of bases, S the set of services and $S(b)$ the set of available services on a base b, where $S(b)$ is equal to S when $b = \perp$.*[4] *A* partial action *p, the formal expression of a user intention, can be modeled as a list of 2-tuples:*

$$p = (e_1, \cdots, e_n) \mid e_i = (b_i, s_i) \text{ with } b_i \in \{\perp\} \cup B; s_i \in \{\perp\} \cup S(b_i) \quad \forall i : (b_i \neq \perp) \vee (s_i \neq \perp) \tag{1}$$

Definition 2 (Service Graph). *Let $E = \{\epsilon = (b, s) \mid b \in B, s \in S(b)\}$ be the set of all available services in a pervasive service environment and $I(E) \subseteq E \times E$ the set of all possible service interactions within this environment.*[5] *Then we can define a service graph in a part $\mathcal{E}$ of the pervasive service environment as*

$$G_{\mathcal{E}} = (\mathcal{E}, \mathcal{I});\ \mathcal{E} \subseteq E;\ \mathcal{I} \subseteq I(\mathcal{E}) \tag{2}$$

The set Γ_E of all valid service graphs in E is defined as:

$$\Gamma_E = \{(\mathcal{E}, \mathcal{I}) \mid (\mathcal{E} \subseteq E, \mathcal{I} \subseteq I(\mathcal{E}))\} \tag{3}$$

Definition 3 (Connected Service Graph). *A service graph $g = (\mathcal{E}, \mathcal{I});\ \mathcal{I} \subseteq I(\mathcal{E})$ is called* connected *iff*

$$\forall \epsilon_0, \epsilon_n \in \mathcal{E},\ \epsilon_0 \neq \epsilon_n : (\epsilon_0, \epsilon_n) \in \mathcal{I} \vee (\exists \epsilon_1, \ldots, \epsilon_{n-1} : (\epsilon_i, \epsilon_{i+1}) \in \mathcal{I};\ (i = 0, 1, \ldots, n-1)) \tag{4}$$

[3] To simplify the model, we do not consider attributes in the following section, they can be easily added lately.

[4] $\perp$ represents "undefined".

[5] The interoperability between services is not studied here.
We assume that $(\epsilon_1, \epsilon_2) \in I(E) \Leftrightarrow ((\epsilon_1, \epsilon_2) \in E \times E) \wedge (\epsilon_1$ is interoperable with $\epsilon_2)$.

Definition 4 (Solution). *A graph* $g_p^E = (\mathcal{E}, \mathcal{I})$; $\mathcal{E} \subseteq E, \mathcal{I} \subseteq I(\mathcal{E})$ *is called* solution *for a given partial action* p *in a pervasive service environment* E *iff*

$$g_p^E \in \Gamma_E \tag{5}$$

$$g_p^E \text{ is connected} \tag{6}$$

$$\forall e = (b, s) \in p \; \exists \epsilon = (\beta, \sigma) \in \mathcal{E} \text{ with } \begin{cases} \beta = b & \text{if } s = \perp, \\ \sigma = s & \text{if } b = \perp, \\ (\beta = b) \wedge (\sigma = s) & \text{otherwise.} \end{cases} \tag{7}$$

Definition 5 (Action Graph). *Let* $\mathcal{S}_p^E$ *be the set of all solutions for* p *in a pervasive service environment* E. *An* action graph $A_p^E \in \Gamma_E$ *of a partial action* p *in the pervasive service environment* E *is a connected service graph containing all solutions for* p.[6]

$$A_p^E = \Big(\bigcup_{(\mathcal{E},\mathcal{I}) \in \mathcal{S}_p^E} \mathcal{E}, \bigcup_{(\mathcal{E},\mathcal{I}) \in \mathcal{S}_p^E} \mathcal{I} \Big) \tag{8}$$

Definition 6 (Complete Action). *Let* $C(g, \gamma)$ *be the function calculating the cost of a solution* g *when it is executed in a context* γ. *Then the* complete action c_p^E *for a given partial action* p *in a service environment* E *is defined as:*

$$C(c_p^E, \gamma) = \min\{C(g_p^E, \gamma) \mid g_p^E \in \mathcal{S}_p^E\} \tag{9}$$

An algorithm using these definitions and declarations to derive the best solution for a given partial action p would be straight-forward, but unfortunately with an exponential complexity. This is not applicable in larger pervasive service environments, so we developed an heuristic approach, solving the query in polynomial time (Alg. 1), whereas $H(p, E, \gamma)$ represents the function fetching from the execution history the complete action for a given partial action p and a pervasive service environment E in a context γ. The approach bases on the following coloring of the nodes, defining a partition of E:[7]

- *Black nodes* ϵ_B: $\forall \epsilon_B = (\beta, \sigma) : \exists (\beta, \sigma) \in p$
- *Red nodes* ϵ_R: $\forall \epsilon_R = (\beta, \sigma) : \exists (\beta, \perp) \in p \vee \exists (\perp, \sigma) \in p$
- *White nodes* ϵ_W: all other nodes

The solution we want to find includes all black nodes, some of the red nodes (so that there is for every $e \in P$ an appropriate node in the solution) and maybe some white nodes. To find the "best" red nodes, we use the following heuristic:

First we calculate using the Shortest Path Algorithm of Dijkstra [3, p. 204] for each red node ϵ_R the distance to all black nodes ϵ_B by using a modified cost function $C_I((\epsilon_R, \epsilon_B), \gamma)$. Beginning with the red node of the shortest distance,

[6] In some rare cases (when $\forall (b, s) \in p : \; b = \perp$) it can happen, that A_p^E is not unique, but this does not imply the algorithm presented below.

[7] Each node wears just one color, a hierarchy is given as: *Black* > *Red* > *White*, at least one black node is required.

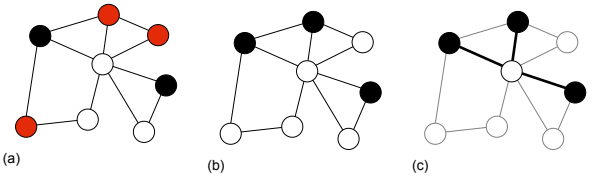


Fig. 2. In this example, the partial action p contains three elements. Two of them define exactly a corresponding (black) node in the action graph and one matches to three different (red) nodes (a). From the red nodes, the one with the nearest distance to a black node is selected and colored black, the others are colored white (b), and the Minimum Cost Spanning Tree containing all black nodes is calculated (c).

we examine all red nodes in order of their distance: If there is a corresponding query part e in the partial action p, which is not already covered by any black node, we color the node black. We repeat this, until every part of p is covered with a black node. The remaining red nodes are colored white. Then, using the Minimum Cost Spanning Tree Algorithm [3, p. 208] we extract from $(E, I(E))$ the graph containing all black nodes as a complete action (see Fig. 2).

5 Evaluation Metrics for a Pervasive Service Environment

To achieve the goal of seamless integration of the pervasive computing system into the user's everyday life environment, it is important that the user percept nearly no delay between expressing his intention and receiving the result of it by the executed action. This leads to the goal of short *execution time* for any action algorithm. This is directly related with the *user satisfaction*, the most important aspect which is nevertheless difficult to measure. The execution time of the transforming process from a partial action to a complete action is just little connected with the complexity of the algorithm, in a pervasive network with a couple of services data transmission costs are much more important. This defines the challenge of minimizing the *network data exchange*, viz minimizing the *size of transferred service descriptions* and the *transmission distance*. All these parameters should not blow the execution of the translating process up when increasing the number of known services and the length of the requests, that means *scalability* as a development goal. Other important issues, when implementing the algorithms on a smart device like a PDA or a wristwatch are the *minimization of CPU-usage* to save energy resources as well as minimizing the *memory usage* to save little memory resources.

6 Related Work

To execute pervasive applications in a "user-aware" way has been worked out in [4] and [5]. Similar as PerSE does, El-Kathib et al. design in [6] a platform trying to maximize user satisfaction while adapting the content of multimedia data

Algorithm 1 Heuristic Translation Algorithm

1: $c = H(p, E, \gamma)$ *// try to find a solution in history*
2: **if** $c \neq \perp$
3: $c_p^E = c$
4: **else**
5: list F *// black nodes*
6: list R *// red nodes*
7: **for each** $\epsilon = (\beta, \sigma) \in E$
8: **if** $\exists e = (b.s) \in p : (b = \beta) \wedge (s = \sigma)$
9: push(F, ϵ)
10: **else if** $\exists e = (b.s) \in p : ((b = \beta) \wedge s = \perp) \vee ((s = \sigma) \wedge b = \perp)$
11: push(R, ϵ)
12: array D *// calculate the minimal distance*
13: **for each** $\epsilon_f \in F$
14: **for each** $\epsilon_r \in R$
15: $d =$ length(dijkstra($(E, I(E))$,ϵ_f,ϵ_r))
16: **if** $d < D[\epsilon_r]$
17: $D[\epsilon_r] = d$
// elements of partial action covered only by red nodes:
18: $p' = \{e = (b, s) \in p \mid (b = (\perp) \vee (s = \perp) \wedge (\nexists \epsilon_f = (\beta, \sigma) \in F : (b' = \beta) \vee (s' = \sigma))\}$
19: sort(R,D) *// sort red nodes by distance*
20: **for each** $\epsilon_R = (\beta, \sigma) \in R$
21: **if** $\exists e = (b, s) \in p' : (b = \beta) \vee (s = \sigma)$
22: push(F, ϵ_r) *// color the node black*
// removed satisfied parts from the partial action:
23: $p' = p' - \{e' = (b', s') \in p' \mid (b' = \beta) \vee (s' = \sigma)\}$
24: $c_p^E =$ MCST($(E, I(E))$,F) *// calculate Min. Cost Spanning Tree on black nodes*

with a dynamically estimated path of enchained transcoders. His solution also relies on graph base composition, where he does not present a formal language to define the adaptation requests. Where it is currently optimized for fixed network structures like available in the Internet, the platform is already resilient against service failures and hereby maybe as well applicable in a pervasive context.

Other user-oriented systems for managing pervasive environments have been developed, for instance Gaia [7] or Aura [8]. Gaia proposes a programming language to construct executable tasks based on the interoperability of services, whereas Aura tries to avoid any interaction with the user and does not present a model for user intention.

In [9], M. Valle et al. introduce a system for dynamic service composition in intelligent environments: they are following a related way as PerSE does, from a partial action (what they call *abstract plan*) through a composition algorithm to a concrete action, in their words *detailed plan*. They have worked out well the mechanisms for service descriptions and service composition, while they do not present a formal way to express intuitive and computer-interpretable user

intention in form of a partial action. They rely on there part on a library of predefined abstract plans, which can be interpreted as a kind of predefined execution history, but this history is not taken directly into account when composing services.

C. Linnhoff-Popien et al. present in [10] a language to describe service request in computer networks. This language specifies in details the semantic of a requested service but has not the intention to support service enchainment as multistep action in a pervasive service environment. The Human-Markup-Language (HumanML) [11] tries to offer an XML-based description for human interacting, but it does not focus on using services in a pervasive environment. Therefore it cannot be used as a replacement for PsaQL, but nevertheless act as user input for creating the partial action.

Another widely explored field of research inspiring the development of this work are Semantic Web Services, as presented for instance by [12]. Ontologies as defined by OWL-S [13] can help to create correspondent and valid action graphs and maximize the user satisfaction with a calculated solution. Semantic composition of Web Services is as well introduced by [14], [15], [16], and [17]. The general challenge of matching several demands semantically on web resources based on their descriptions is treated by [18].

A pervasive environment will be characterized by the availability of application context information [19]. PerSE will use the contextual information to build an appropriate action graph and to select the best solution as complete action. Earlier approaches of introducing context based adaption into pervasive environments are presented in [20] and [17]. When one wants to use public available services in a pervasive network seamlessly, the security is one of the main points. First attempts are made by introducing smart authentication into PerSE like worked out in [21].

7 Conclusion and Open Issues

This paper has presented a strategy and a methodology to take the user intention into account when composing service-based actions in a pervasive service environment. It introduced PsaQL, a language to express user intention in a pervasive service environment. We outlined algorithms extending this partial action to an executable graph of services using the service descriptions, the context information and the execution history. The next step will be an implementation of the translation service as prototype and the evaluation of this prototype in an appropriate environment.

Following this research work, a couple of issues remain open to be developed in future time. Capturing the user intention using PsaQL will not stay the last conclusion of wisdom, more seamless ways like voice or gesture recognition will be developed. Independent of the kind of expression the user selects to communicate his intention, PsaQL will be an easy interface between the intention capturing module and the module developing the complete action to execute.

References

1. Weiser, M.: The Computer for the 21st Century. Scientific American (September 1991) 94–104
2. Date, C.J.: A guide to the SQL standard. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1986)
3. Manber, U.: Introduction to Algorithms: A Creative Approach. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1989)
4. Sousa, J.P., Garlan, D.: Improving user-awareness by factoring it out of applications. In: UbiSys'03 - System Support for Ubiquitous Comp. Workshop. (2003)
5. Saif, U., Pham, H., Paluska, J.M., Waterman, J., Terman, C., Ward, S.: A case for goal-oriented programming semantics. In: UbiSys'03 - System Support for Ubiquitous Computing Workshop. (2003)
6. El-Khatib, K., v. Bochmann, G., Saddik, A.E.: A qos-based framework for distributed content adaptation. In: Quality of Service in Heterogeneous Wired/Wireless Networks, QSHINE 2004. (2004) 308–312
7. Román, M., Hess, C., Cerqueira, R., Ranganat, A., Campbell, R.H., Nahrstedt, K.: Gaia: A middleware infrastructure to enable active spaces. IEEE Pervasive Computing (2002) 74–83
8. Garlan, D., Siewiorek, D., Smailagic, A., Steenkiste, P.: Project aura: Toward distraction-free pervasive computing. IEEE Pervasive Computing **1** (2002) 22–31
9. Vallée, M., Ramparany, F., Vercouter, L.: Composition flexible de services d'objets communicants. In: UBIMOB 05. (2005)
10. Popien, C., Meyer, B.: A service request description language. In: FORTE'94. Chapman & Hall, Bern (1994) 14–32
11. Best, K.F.: Oasis standards work. Markup Lang. **3** (2001) 241–249
12. McIlraith, S.A., Son, T.C., Zeng, H.: Semantic web services. IEEE Intelligent Systems **16** (2001) 46–53
13. McGuinness, D.L., van Harmelen, F.: OWL Web Ontology Language Overview. online: http://www.w3.org/TR/2004/REC-owl-features-20040210/ (2004)
14. Staab, S., van der Aalst, W.M.P., Benjamins, V.R., Sheth, A.P., Miller, J.A., Bussler, C., Maedche, A., Fensel, D., Gannon, D.: Web services: Been there, done that? IEEE Intelligent Systems **18** (2003) 72–85
15. Sheshagiri, M., desJardins, M., Finin, T.: A planner for composing services described in DAML-S. In: Proceedings of ICAPS'03 Workshop on Planning for Web Services. (2003)
16. Sycara, K.P., Paolucci, M., Ankolekar, A., Srinivasan, N.: Automated discovery, interaction and composition of semantic web services. J. Web Sem. **1** (2003) 27–46
17. Vukovic, M., Robinson, P.: Adaptive, planning-based, web service composition for context awareness. In: Second Int. Conference on Pervasive Computing. (2004)
18. Noia, T.D., Sciascio, E.D., Donini, F.M., Mongiello, M.: A system for principled matchmaking in an electronic marketplace. In: WWW. (2003) 321–330
19. Ma, J., Yang, L.T., Apduhan, B.O., Hunag, R., Barolli, L., Takizawa, M.: Towards a smart world and ubiquitous intelligence: A walktrough from smart things to smart hyperspaces and ubickids. In: International Journal of Pervasive Computing and Communications. Volume 1. Troubador Publishing Ltd. (2005) 53–68
20. Ranganathan, A., Campbell, R.H.: An infrastructure for context-awareness based on first order logic. Personal and Ubiquitous Computing **7** (2003) 353–364
21. Saadi, R., Pieson, J.M., Brunie, L.: APC: Access pass certificate. distrust certification model for large access in pervasive environment. In: IPCS'05. (2005)

The Performance Estimation of the Situation Awareness RFID System from Ubiquitous Environment Scenario

Dongwon Jeong[1], Heeseo Chae[2], and Hoh Peter In[2]

[1] Dept. of Informatics & Statistics, Kunsan National University,
San 68, Miryong-dong, Gunsan, Jeollabuk-do, 573-701, Korea
djeong@kunsan.ac.kr
[2] Dept. of Computer Science & Engineering, Korea University,
1, 5-ka, Anam-dong, Sungbuk-ku, Seoul, 136-701, Korea
{royalhs, comtaek, hoh_in}@korea.ac.kr

Abstract. Many sensors providing situation data will be in everywhere under the ubiquitous environment. It requires the current RFID system should be extended to recognize and use situation information from the sensors. We already proposed a new RFID system architecture that is suitable for the coming ubiquitous environment, and basically consists of the key components such as the inference engine, use policy, and definition language. It has four alternatives (types) depending on the role distributions to the components. This paper shows the performance evaluations of the alternatives. Using the research results, we can select a proper RFID system architecture for corresponding ubiquitous application. The contributions of this paper are as follow: providing a new RFID system architecture to recognize, analyze, and utilize the situation information; defining the four alternatives for ubiquitous applications.

1 Introduction

In the ubiquitous computing environment, the computing paradigm as well as our life style will be changed. Some people can get the real-time transportation information that the next bus will arrive soon at the bus stop. Other people are provided seamless movies or high quality games when they move from outdoor with mobile phone to indoor with more efficacious resource like TV, PC, and so on.

Radio Frequency Identification (RFID) system and Situation-Aware (SA) technology are center of these transition. One of the most popular ubiquitous applications is the RFID system. Currently, the RFID system is studying vigorously. The RFID system generally consists of RFID tag, RFID reader and Data Processing Subsystem (DPS). The RFID tag saves information and The RFID reader reads tag information. Then the DPS treats these data and uses the processed information. The RFID system has limitation that this system mostly used to discriminate an individual in limited environment, but it attentions its broad appliance. Especially, the coming ubiquitous environment requires the enhancement of the current RFID system [9].

Situation-Aware (SA) technology supports that devices are able to take actions automatically and timely depending on situation information offered by surrounding

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 987 – 995, 2005.
© IFIP International Federation for Information Processing 2005

sensor. It is based on the existing sensor-network. However, the SA technology is more aware of situation and its processes are performed intelligently with a variety of information collected by sensors, not at a certain time sequence.

In [9], we defined and proposed a RFID system architecture which has been integrated to conquer the limitations of RFID system over the SA technology. [9] just showed the new RFID system architecture and its system alternatives. For its appliance, the differences among the alternatives should be studied. The research result can be used as templates to select proper system architectures for target applications. Most of all, performance efficiency of each system architecture type (alternative) should be considered for actual application. Therefore, this paper focuses on the performance evaluation. With the evaluation result, we can choose the most suitable architecture.

2 SA-RFID System Overview

In [9], the SA-RFID system was proposed for the current RFID system to be suitable for the ubiquitous computing environment. The SA-RFID system enhances usability of the current RFID system by use of information acquired from various sensors.

The proposed SA-RFID system is capable of recognizing such a situation and determining the use of the resource based on the recognized situation information. The system architecture is extended from the current RFID system architecture.

[9] distributed the SA-RFID system into four types. The conceptual architecture is illustrated in Fig. 1.

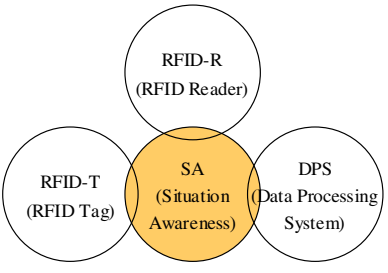


Fig. 1. Conceptual architecture of SA-RFID system

In Fig. 1, a RFID tag (RFID-T) holds ID information of an object, and it is the same as the traditional-current RFID-T function. The RFID reader (RFID-R) reads information from the RFID-T. In addition, in case of the SA-RFID system architecture, the RFID-R can provides extended functions that can acquire situation information from sensors, infer proper actions from the situation information, and execute the actions.

The data processing system (DPS) basically utilizes the information acquired by the RFID-R. The DPS also contains information pertaining to the use policies and determines suitable actions by inferring and judging the situation information.

For supporting the SA technology, the RFID system requires functions to collect situation information, define use policies with a profitable representation language, and infer for determining valid actions using the use policies and the situation information collected. We satisfy these requirements by applying the concept of the SA.

Fig. 2 depicts the types of the SA-RFID system architectures classified based on functions for supporting SA (Situation Awareness) concept and related operations [9].

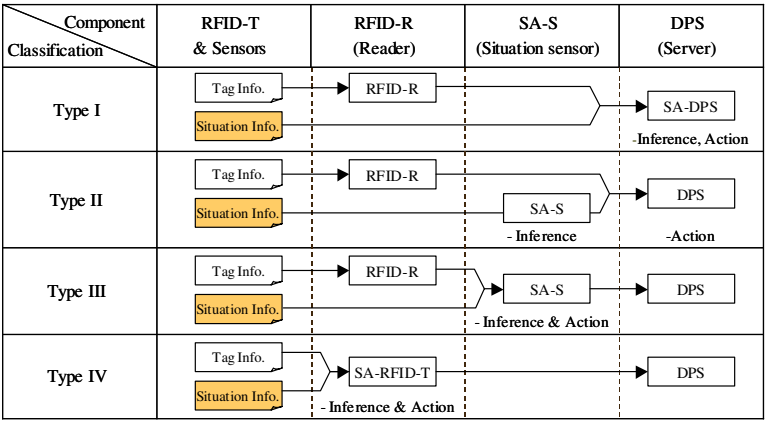


Fig. 2. Classification of SA-RFID system architecture

The architectures are classified based on whether a new component is added into the existing RFID system. In other words, the architectures are grouped into a Non SA-Sensor Based (NSSB) architecture not requiring the new component SA-Sensor and a SA-Sensor Based (SSB) architecture requiring the new component. In the NSSB architecture, the SA inference and execution are processed by the DPS (Type I) or by the RFID-R (Type IV). In the SSB architecture, the situation information is inferred by the SA-Sensor but the execution according to the use policies is processed by the SA-Sensor (Type II) or by the DPS (Type III).

As for Type I which extends the functions of the DPS, the situation information is received directly from the sensors, and the DPS infers the situation information based on the received situation information and tag information acquired from the RFID-R and executes the resulting actions referred with the use policies.

As for Type II, the new component SA-Sensor is added and forwards to the DPS the situation information collected from the sensors. The SA-Sensor infers the situation information from the sensors, and sends the results to the DPS. The DPS obtains the tag information from the RFID-R and the actions from the SA-Sensor, and then executes the actions.

Type III is similar to Type II, the SA-Sensor processes both of the inference and execution depending on the use policies. And the SA-Sensor gets the tag information and sends it to the DPS with the action result.

Finally, Type IV is an extended architecture and the RFID-R performs the functions of the SA-Sensor. Therefore, the RFID-R processes the inference and execution by collecting the situation information from the sensors.

3 States and Transition in Architecture Alternatives

Type I: SA-DPS based system
The SA-DPS based system has not a concept being dependent of SA part and DPS part but a concept of architecture extended and integrated. This architecture can be realized through easy and simple extension of the existing DPS. However, it is a disadvantage to be delayed in gathering information by remote communication between sensors and the SA-DPS so that reliability could not be assured.

On SA-DPS based system, the RFID-R sends the information came from the RFID-R to SA-DPS and simultaneously sends situation information received from neighbor sensors to the SA-DPS as well. Fig. 3 shows the sequence diagram model of the SA-DPS based RFID system. The sequence diagram model is widely being used for the time-based process design [12].

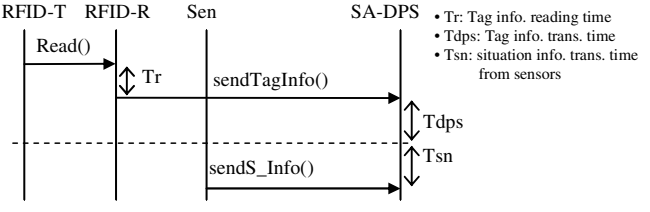


Fig. 3. SA-DPS Based RFID System sequence diagram

We assume the data transmission time and processing time is uniform. The architecture's performance can be evaluated in terms of processing some portion of data by the summation of *Tdps*, which is the time that RFID-R spends sending the information to SA-DPS, and the time that gathered information from sensors reaches SA-DPS. The calculation model for the whole processing time of the SA-DPS based RFID system $T_{SA\text{-}DPS}$ is defined as follows:

$$T_{SA\text{-}DPS} = Tdps + Tr + Tsn \qquad (E1)$$

As you can see a relative gap of delayed time in Fig. 4, the point which we have to pay attention to is that there is an issue for spatial heterogeneousness between sensors gathering situation information and SA-DPS receiving it. Hence, the problem from remote communication between the sensors and SA-DPS can be solved when all of sensors can make it with securing communication reliability.

Type II: SA-Sensor supplementary system
The SA-Sensor supplementary RFID system architecture requires an additional sensor SA-S (Situation Aware-Sensor) which plays roles to gather situation information from sensors and send the DPS the acquired situation information. In this architecture type, while the RFID-R sends the tag data came from the RFID-T to the DPS, the SA-sensor sends the information gathered from sensors to the DPS at once with directing them. The tag data and situation information pushed to the DPS help the system make a decision for usage policy and run it. Fig. 4 shows the sequence diagram of this system architecture.

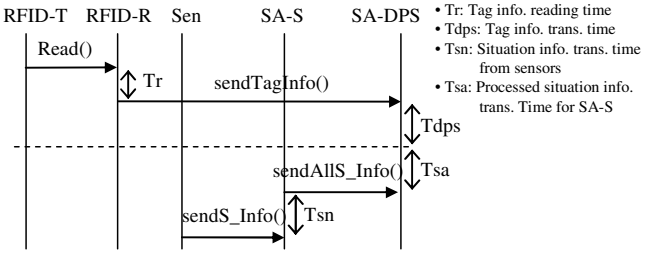


Fig. 4. Sequence diagram for the SA-Sensor supplementary RFID System

We assume that the data transmission time and the data processing time in a terminal machine like other architectures, and particularly there is no difference between each of spending time in sending data from senor to SA-S and from SA-S to DPS.

Let the whole processing time of the SA-Sensor supplementary RFID system (the processing time and data transmission time) for evaluating performance is $T_{Supplementary}$. We can define $T_{Supplementary}$ as follows:

$$T_{Supplementary} = Tdps + Tr + Tsa + Tsn \quad (E2)$$

Thus, the SA-Sensor supplementary architecture adopted SA-Sensor can improve the reliability of communication between sensors because reading the RFID tag data and gathering situation information from sensors are addressed in same space. Nevertheless, complexity of the architecture is still a problem.

Type III: SA-Sensor oriented system

On the SA-Sensor oriented architecture, SA-Sensor plays an intensive role. In the SA-Sensor supplementary system architecture, the SA-Sensor delivers it to the DPS after gathering the information from sensors. However, the strengthened SA-Sensor sends just only processed and inferred situation information to the DPS.

Fig. 5 expresses the sequence diagram for operations of this type. This figure shows sequentially key processing steps in the SA-Sensor oriented RFID system. With this diagram in Fig. 5, we can define the calculation model for estimating the performance of the SA-Sensor oriented RFID system. Let the whole processing time of the SA-Sensor oriented system is $T_{oriented}$. Then $T_{oriented}$ is defined as follows:

$$T_{oriented} = Tsas\text{-}r + Tr + Tsa + Tsn \quad (E3)$$

However, in the equation above, the process converting the data came from sensors to situation information is missed, in other words, the data processing time is not considered. In the case of architecture Type I, even though the time for processing situation information is not considered due to system characteristics of the SA-DPS, in the case of SA-Sensor oriented system which has a function for situation inference, a factor explaining data processing time is inserted for accurate performance evaluation. The equation for SA-Sensor oriented RFID system $T_{oriented}$ can be redefined as follows:

$$T_{oriented} = Tdps + Tr + Tsa + Tsn + Ts\text{-}op \quad (E4)$$

Therefore, the SA-Sensor oriented system improved especially in the function of the SA-Sensor makes reliability of information communication increased because of short distance communication between sensors and readers, but lots of resources and computation power are required due to the functions added to the SA-Sensor.

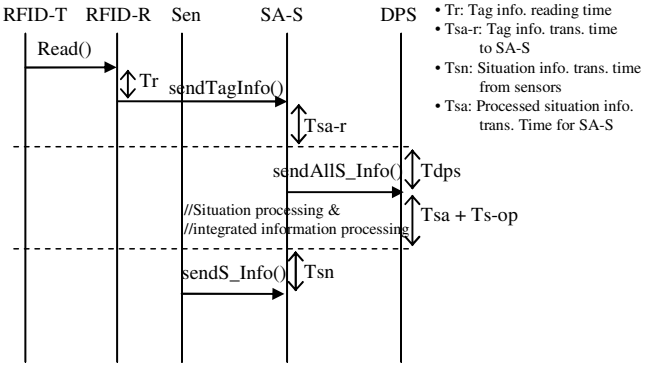


Fig. 5. SA-Sensor Oriented RFID System sequence diagram

Type IV: SA-RFID reader based system

Type IV is an extended architecture especially in terms of reader's function on other previous architecture. The reader plays a role not only gathering tag data but also gathering raw data from sensors, and also processes the obtained data (including the raw data) and finally sends the processed and integrated situation information to the DPS. We can call the RFID reader having such functions as SA-RFID/SA reader.

This architecture vests the reader in traditional RFID system with the additional functions. It can keep consistency of information management because the behaviors reading tag data and inferring situation information are dealt with in a component at once. It takes little delay time to move on to when considering performance evaluation on SA-RFID reader based system because the architecture of the system has low complexity comparing with other types.

Fig. 6 illustrates the sequential operation steps of this system as a sequence diagram model. According to the diagram, the equation for calculating the performance of this SA-RFID reader based system $T_{SA\text{-}Reader}$ is as follows:

$$T_{SA\text{-}Reader} = Tr + Tdps + Tsn \quad (E5)$$

However, overall performance of the system depends on the time consumption factors when reading tag data in the SA-RFID, gathering sensed data, and inferring situation information. Therefore, the above equation can be redefined as follows:

$$T_{SA\text{-}Reader} = Tr + Tdps + Tsn + Tr\text{-}op \quad (E6)$$

As shown in Fig. 6, in the process moving from the SA-RFID reader to the DPS, inferring situation information and reading tag data are addressed totally through integrating the two kinds of information.

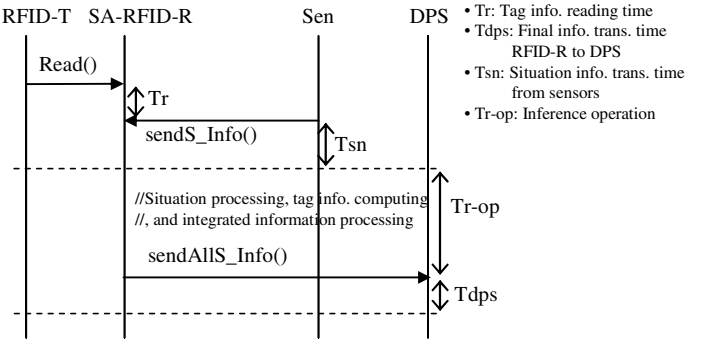


Fig. 6. SA-RFID reader based RFID system architecture

4 Evaluations

In this paper, SmartView performance is scenario to which we will pay attention and apply SA-RFID. SmartView is the situation awareness based multimedia contents provider system in ubiquitous environment which provides multimedia data and games for users anytime, anywhere, so users can enjoy various services along with their situation regardless of devices and circumstances [13].

In SmartView scenario, the process gathering tag information plays a role in identifying which device will be provided with service. Table 1 shows the qualitative comparisons between the system architectures.

Table 1. architecture type estimation over the Smartview scenario

	Architecture type I	Architecture type II
Tag info	Identifying platform	Identifying platform
Situation info.	Providing location info.	Sensing per resource
DPS	Integrated application	Application
Delay time	Most long	Less than I
Estimation	Bad	Not bad
	Architecture type III	Architecture type IV
Tag info	Identifying platform	Gathering integrated info.
Situation info.	Sensing per resource	Gathering integrated info.
DPS	Application	Integrating sensed data
Delay time	Less than I	Most short
Estimation	Good	Excellent

Fig. 7 shows the evaluation result with this scenario (These values which are used in estimation are actual value of real system) [10, 11].

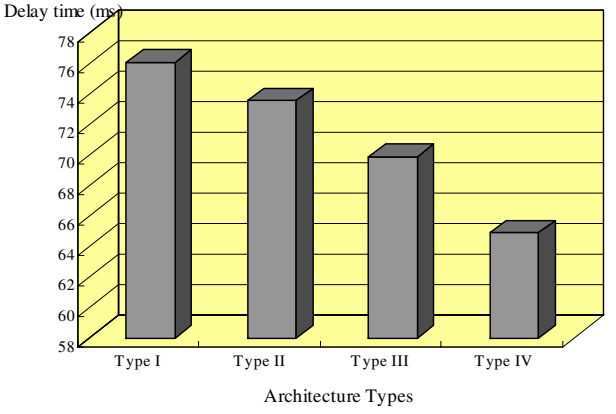


Fig. 7. SmartView Performance Estimation

Consequently, the results of estimation with Table 1 and Fig. 7 can be explained as follows:

- Delay time : IV < III ≤ II < I
- Performance capability : IV > III ≥ II > I

5 Conclusion

Ubiquitous computing is considered as the next generation computing paradigm. Many sensors will be able to provide various and abundant situation information in the ubiquitous computing environment. Such a change of computing paradigm and development of technologies will require more extensive and far-reaching usability than the current RFID systems. Currently, the functions of the current RFID system are limited only to identification and recognition of objects. Thus, the systems provide the functions for simply reading and processing ID information in little consideration of situations.

In case that information on a RFID tag is used only within a specific time or at a specific place, current RFID system architecture does not consider such a use policy at all. To support this application, it is sometimes needed to receive information from sensors detecting location information and to determine availability of an object identified from the tag at a specific time. Valid geographic location information should be received from the sensor for the access permission.

Therefore, a new extended RFID system architecture should be defined for determining and utilizing various use policies for the RFID system based on information acquired from the diverse sensors. The definition of the extended RFID system architecture requires methods to interpret and define the situation information, and also needs to consider how to manage the information from the various sensors.

In this paper, we proposed the use policies and a formal language for the expression of the use policies to support the SA concept in the ubiquitous computing environ-

ment, suggested several types of SA-RFID system architecture. Finally, we qualitatively compared and evaluated the system architectures.

Our system architectures may be selectively utilized as need. It is expected that our architectures overcome simple and restrained functions of the existing RFID system and provide far-reaching applications. Above all, it is feasible to utilize the various RFID tags based on the situation information and determine the diverse access methods.

Further researches are demanded on implementation technologies of optimum inference engines. Especially, the compactness of the inference engines allows the definition on more diverse system architectures beyond our system architectures.

Acknowledgement

Hoh Peter In and Dongwon Jeong are the corresponding authors.

References

1. K. Finkenzeller, “RFID Handbook: Radio-Frequency Identification Fundamentals and Applications,” John Wiley & Sons, (2000)
2. S. E. Sarma, S. A. Weis, D. W. Engels, “RFID Systems and Security and Privacy Implications,” Lecture Notes in Computer Science, In Workshop on Cryptographic Hardware and Embedded Systems, LNCS 2523, February (2002) 454-470
3. S. Yau, F. Karim, Y. Wang, B. Wang, and S. Gupata, “Reconfigurable Context-Sensitive Middleware for Pervasive Computing,” IEEE Pervasive Computing, Vol. 1, No. 3, September (2002) 33-40
4. S. S. Yau and F. Karim, “Adaptive Middleware for Ubiquitous Computing Environments,” Proc. IFIP 17th WCC, August (2002)
5. S. Yau, Y. Wang, and F. Karim, “Developing Situation-Awareness in Middleware for Ubicomp Environments,” Proc. 26th Int'l Computer Software and Applications Conference (2002) 233-238
6. P. J. Daugherty, T. P. Stank, and D. S. Rogers, “Third-Party Logistics Service Providers: Purchasers' Perceptions,” The Journal of Supply Chain Management, Vol. 32, No. 2 (1996) 23-29
7. Association for Automatic Identification and Mobility (AIM), http://www.aimglobal.org/
8. H. Wang, M. K. O. Lee, and C. Wang, “Consumer Privacy Concerns about Internet Marketing,” Communication of the ACM, Vol.41, No.3 (1998) 63-70
9. D. Jeong, Y.-G. Kim, and H. P. In, “New RFID System Architectures Supporting Situation Awareness under Ubiquitous Environments,” Science Publications, Journal of Computer Science, Vol. 1, No. 2 (2005) 114-120
10. J. J. Chen and C. Adams, “Short-range Wireless Technologies with Mobile Payments Systems,” ACM Press, Sixth International Conference on Electronic Commerce (ICEC’04), Delft, The Netherlands, October (2004) 649-656
11. Intermec,“IP3 Portable RFID(UHF) Reader,” (2005), http://www.intermec.com/eprise/main/Intermec/Content/Products/Products_ShowDetail?Product=RFID2_IP3#
12. Object Management Group (OMG), “UML Specifications 2.0,” October (2004)
13. H. P. In, C. Kim, and S. S. Yau, “Q-MAR: An Adaptive QoS Management Model for Situation-Aware Middleware,” EUC2004, Aizu-Wakamatsu City, Japan, August (2004) 972-981

The Content Analyzer Supporting Interoperability of MPEG-4 Content in Heterogeneous Players

Hyunju Lee and Sangwook Kim

Department of Computer Science, Kyungpook National University,
Sankyuk-dong 1370, Bukgu, Daegu, Korea
{hyunju, swkim}@cs.knu.ac.kr

Abstract. MPEG-4 supports object-based interactive multimedia applications. To compose complex multimedia scene, MPEG-4 supports BIFS and XMT as scene description. For interoperability of the MPEG-4 content and object-based rendering information management, this paper proposes the content analyzer that supports multiple scene description. In order to improve content management's efficiency, the content analyzer provides module optimization from scene description's parsing to scene information management. This analyzer supports exact analysis of the scene description and object-based core information management. And it can be applied to XML-based heterogeneous players.

1 Introduction

MPEG-4(Moving Picture Experts Group, ISO/IEC 14496) [1] targets interactive multimedia applications and also offers content-based functionality, i.e. the possibility of accessing and manipulating individual objects in the scene.

MPEG-4 describes the composition of a complex multimedia scene which relies on the concept of both BIFS and XMT as scene description. The scene description describes the spatial-temporal composition of objects in a scene and provides media data to the presentation layer of the MPEG-4 terminal. The BIFS [2] is a binary format of VRML-based, and it is usually used when MPEG-4 content is played in the MPEG-4 player. In contrast, the XMT [3] is a XML-based textual format. The main advantage of XMT is exchangeability of content and interoperability with other media players of XML-based. Other existing MPEG-4 players focus on the BIFS as scene description. Therefore, if MPEG-4 content contains XMT as content's scene description, MPEG-4 content can not be reused by other purpose.

In this paper, we propose the content analyzer that supports two types of MPEG-4's scene description sufficiently and it also supports description conversion for interoperability of content. It consists of four core components: the description separator, the BIFS/XMT engine and the information manager. To improve content management's efficiency, the content analyzer provides module optimization from scene description's parsing to scene information management. To expand module easily and improve throughput of player, each module in content analyzer is independently managed. Specially, the XMT engine is adaptively designed for scene description change.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 996 – 1005, 2005.
© IFIP International Federation for Information Processing 2005

This paper is organized as follows. Section II, we propose the content analyzer in MPEG-4 player. In section III, presents experimental results. Finally, section IV presents conclusions.

2 The Content Analyzer

The content analyzer supports two types of MPEG-4's scene descriptions that are defined in MPEG-4 standard sufficiently.

In this chapter, we describe the content analyzer in detail.

2.1 The Content Analyzer Architecture in MPEG-4 Player

The MPEG-4 player's action processing is as follows. The MPEG-4 file formatter analyzes the header of an MPEG-4 file, which separates several media data that are included in a file and delivers it to the system decoder. The system decoder reads media data in decoding buffer, and decodes according to decoding time. The content analyzer composes a scene tree and arranges data of buffer on screen through the presenter. The player receives various user events in the user interface.

Figure 1 shows the architecture of the content analyzer in MPEG-4 player and the processing step (from scene description's parsing to scene information management) of the content analyzer.

The content analyzer consists of three major components. These are the description separator for analysis and classification of scene description, the BIFS/XMT engine for parsing process, the information manager for independent management of object information according tov object's properties.

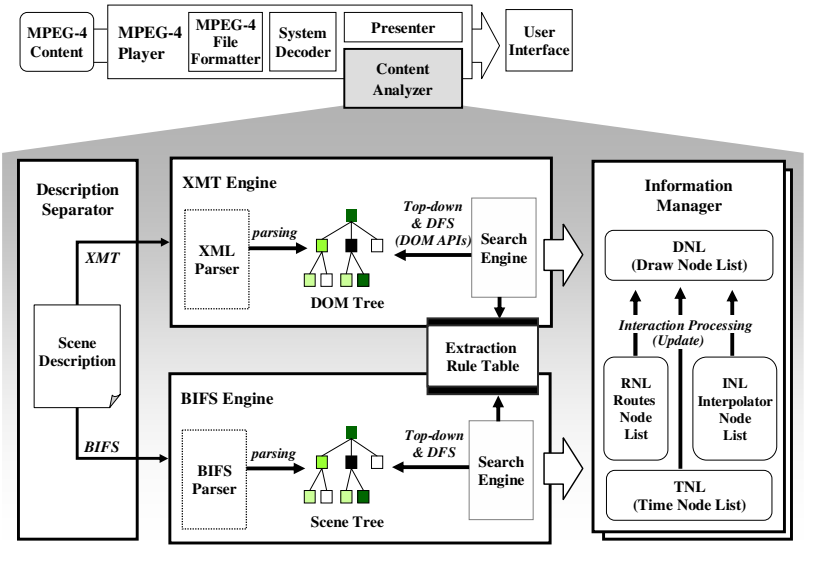


Fig. 1. The architecture of the content analyzer in MPEG-4 player and the processing step

2.2 The Description Separator

The description separator analyzes the scene description that is defined in MPEG-4 content, and calls each engine.

The BIFS's purpose is content playback in MPEG-4 player only. Therefore, it is usually included in MPEG-4 content. But, it is difficult to modify MPEG-4 content because BIFS is binary format. In contrast, the XMT is independently transmitted with MPEG-4 content. The XMT has two creation reasons. First, it supports representation of MPEG-4 scene description using a textual syntax. Therefore, a user can modify media content more easily. Second, it facilitates interoperability with the VRML, the X3D and the SMIL through content exchange. Therefore, if the MPEG-4 content contains an XMT file as scene description, the MPEG-4 content can be played in other XML-based players.

The description separator analyzes two types of scene descriptions exactly, and various analyzed information are used to playback of media data on screen according to use purpose. The description separator has three analysis types according to the content's scene description. First, the description separator calls the BIFS engine when the content contains BIFS only. Second, the description separator calls the XMT engine when the content contains XMT only. Third, the content can include both BIFS and XMT. In this case, the MPEG-4 content can be played to other media players as well as MPEG-4 player. At this time, the description separator calls the BIFS engine and the XMT engine together, and each information manager is created. The player can use parsing result according to contents' playback purpose and playback environment properly.

2.3 The BIFS Engine

If MPEG-4 content contains BIFS as scene description, the MPEG-4 player presents media according to BIFS (BInary Format for Scenes) and OD (Object Descriptor) information. The OD [4-5] identifies and describes elementary streams and associates these streams with corresponding audiovisual scene data. It is used to connect BIFS with media stream resources.

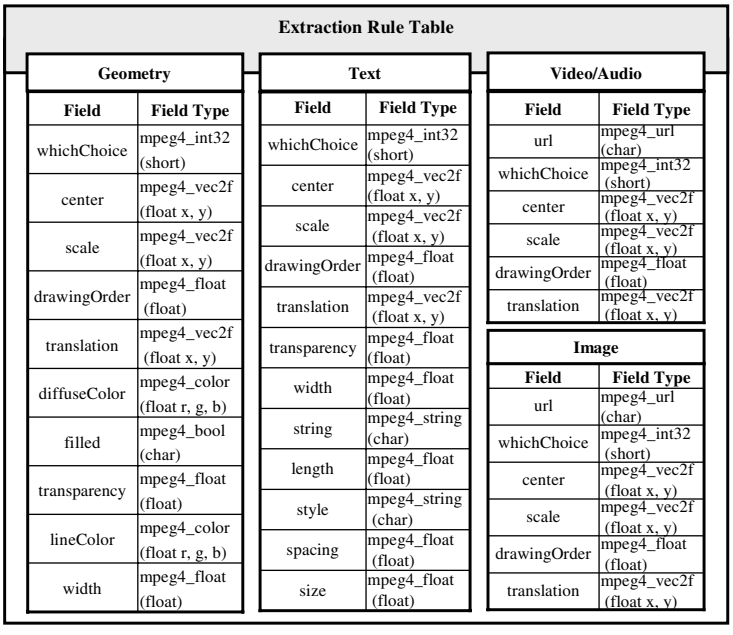
The BIFS engine analyzes the BIFS of MPEG-4 content exactly, searches and extracts the core information of objects that need to rendering. It consists of three components: the BIFS parser, the search engine and the extraction rule table.

The MPEG-4 describes objects and their behavior in hierarchical models. The MPEG-4 uses the concept of a scene tree with object nodes. The scene tree can be obtained through BIFS's parsing and it is used to hierarchically organize and manage the content of scene data. The multimedia information that is defined in scene tree can extract the necessary information through various search mechanisms and manage media data in user's point of view. To support sufficiently object's interactive information that is defined in tree, the MPEG-4 player needs to search the tree more exactly and fast.

In this paper, we use a top-down method and DFS (Depth First Search) mechanism for object-based search. The search engine receives a scene tree's header, searches an object node and judges whether searched object node can draw. When the search engine searches an object, we extract object information according to extraction rules.

The extracted object information is stored in the information manager. The fast search mechanism will be described in evaluation results section. The extraction rule table is based on the search engine and the information manager.

Figure 2 shows an example of the extraction rule table.



Extraction Rule Table

Geometry

Field	Field Type
whichChoice	mpeg4_int32 (short)
center	mpeg4_vec2f (float x, y)
scale	mpeg4_vec2f (float x, y)
drawingOrder	mpeg4_float (float)
translation	mpeg4_vec2f (float x, y)
diffuseColor	mpeg4_color (float r, g, b)
filled	mpeg4_bool (char)
transparency	mpeg4_float (float)
lineColor	mpeg4_color (float r, g, b)
width	mpeg4_float (float)

Text

Field	Field Type
whichChoice	mpeg4_int32 (short)
center	mpeg4_vec2f (float x, y)
scale	mpeg4_vec2f (float x, y)
drawingOrder	mpeg4_float (float)
translation	mpeg4_vec2f (float x, y)
transparency	mpeg4_float (float)
width	mpeg4_float (float)
string	mpeg4_string (char)
length	mpeg4_float (float)
style	mpeg4_string (char)
spacing	mpeg4_float (float)
size	mpeg4_float (float)

Video/Audio

Field	Field Type
url	mpeg4_url (char)
whichChoice	mpeg4_int32 (short)
center	mpeg4_vec2f (float x, y)
scale	mpeg4_vec2f (float x, y)
drawingOrder	mpeg4_float (float)
translation	mpeg4_vec2f (float x, y)

Image

Field	Field Type
url	mpeg4_url (char)
whichChoice	mpeg4_int32 (short)
center	mpeg4_vec2f (float x, y)
scale	mpeg4_vec2f (float x, y)
drawingOrder	mpeg4_float (float)
translation	mpeg4_vec2f (float x, y)

Fig. 2. An example of the extraction rule table

In extraction rule table, object and object's main property field are defined, which do playback in player, and is possible to extend according to playback environment. For example, if search engine searches a video object, it extracts property information such as url, scale, translation that is defined to extraction rule table. The player achieves search optimization through extraction rules.

2.4 The XMT Engine

The XMT (eXtensible MPEG-4 Textual format) is a framework for representing MPEG-4 scene description using a textual syntax. The XMT facilitates interoperability with XML-based other applications. The XMT format can be interchangeable among heterogeneous players such as SMIL players, VRML players and MPEG-4 players.

The XMT framework consists of two levels of textual syntax and semantics: the XMT-A format and the XMT-Ω format. The XMT-A is an XML-based version and provides one to one mapping between the textual format and binary format. The XMT-Ω is a high level abstraction of MPEG-4 features designed based on the SMIL. In addition, the XMT-C (Common) section contains the definition of elements and

attributes that are used within either XMT-A or XMT-Ω. The XMT can be analyzed by XML parser if XMT observes XML's basis grammar and grammar of DTD (Document Type Definition) or schema that is defined in document.

In this paper, the XMT engine manages XMT file when MPEG-4 content contains XMT as scene description and it supports both XMT-A and XMT-Ω.

The XMT engine analyzes the XMT of MPEG-4 content exactly, searches the DOM tree and extracts the core information of objects that need to rendering. It consists of three core components: the XML parser, the search engine and the extraction rule table. The DOM tree is created after XMT parsing through the XML parser. It is a data structure used to hierarchically organize and manage the content of scene data. The DOM tree is very similar to scene tree of BIFS in case of media representation.

The search mechanism for DOM Tree of XMT is the same as scene tree of BIFS. We search the DOM tree through top-down and DFS method. Extraction information and search method are the same as scene tree but extraction order has different approach because the DOM tree's content description form differs with the scene tree. The search engine can approach to the DOM tree using various DOM APIs. We extract basic information of access in extraction rule table like the BIFS engine.

2.5 The Information Manager

The main characteristic of MPEG-4 is the object-based coding and representation of an audiovisual scene. The BIFS/XMT engine creates interiorly the tree through scene description parsing. We applied the optimized search technique to the tree. However, a tree is only possible hierarchic management of all object information that is included in MPEG-4 content. For reusability and flexibility of object-based, we propose the information manager that manages object information efficiently. It consists of five data structure for basic object rendering and dynamic scene composition processing of objects. It is adaptively designed for rendering.

For basic management of rendering object, we create the DNL (Draw Node List). The DNL consists of heads and subs. An object consists of a single head and several subs. It is a structure that is added to the head and subs whenever an object is searched. The head contains ObjectID and the sub contains several property nodes.

The scene description's ROUTEs mechanism describes interactivity and behavior of objects. ROUTEs are comprised of connections that assign the value of one field to another field in the tree interiorly. It is information that processes dynamic user events that are defined in the MPEG-4 standard. We create the RNL (Routes Node List) for interaction processing. The RNL consists of SourceObjectID, EventType, TargetObjectID, ActionType and ModificationValues.

The MPEG-4 provides stream animation to the scene. We compose two INLs (Interpolator Node List) for interpolator processing. If an object includes interpolator node, the search engine divides whether interpolator is color or position, and store it in each List (CINL: Color Interpolator Node List, PINL: Position Interpolator Node List). The INL consists of ObjectID, key (time value) and keyValue (changing value according to key).

The MPEG-4's another mechanism for dynamic scene composition is time-related interaction. We create the TNL (Time Node List) for time-related interaction. An

object needs playback start time and end time. The TNL consists of ObjectID, two time values (StartTime and EndTime) and PropertyValue. The PropertyValue is activity or inactivity information of object. The MPEG-4 player preferentially processes time information than other user events. Each object of the first screen is played according to time information that is defined to the TNL, and receives other user events in addition.

Figure 3 shows five data structure of the information manager.

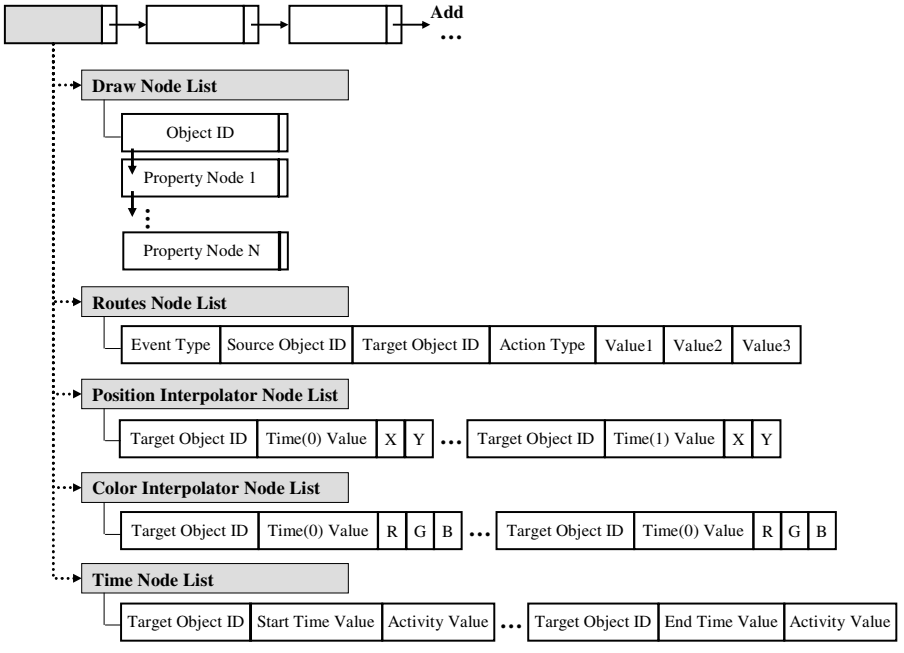


Fig. 3. The information manager

The information manager is dynamically created according to scene description's kind. If MPEG-4 content contains two types of scene description, two information managers are created. The information manager reduces tree re-composition and frequent search of object information because it manages interaction information as well as basic rendering object information by object-based. And, it reduces the load of system-side for additional scene update processing. Finally, whenever scene is updated, it can offer high-level quality of scene to user through reusability of object.

3 Experimental Results

We developed the content analyzer that supports two scene description formats and complete2D profile that is defined in MPEG-4 standard. The proposed content analyzer can manage the MPEG-4 content that consists of various media objects.

3.1 Implementation and Application

In the implementation, we used Xerces-C++ version 2.5.0 as XML parser and XMT API is written in C++. The content analyzer's other modules are written in C/C++ and Visual C++ based on Ms-Windows.

Figure 4 shows two XMT formats and an execution example of MPEG-4 content in the KNU MPEG-4 player. The KNU MPEG-4 Player [6] is an object-based interactive MPEG-4 player which is based on MS-Windows. Two XMT files are matched each schema.

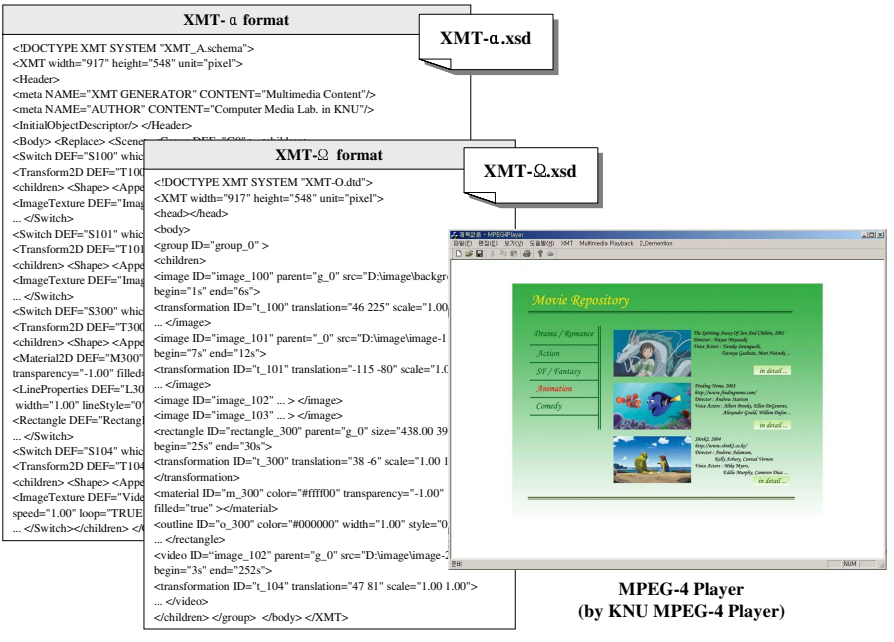


Fig. 4. Two XMT formats and an execution example of MPEG-4 content in the MPEG-4 player

In figure 4, the MPEG-4 content consists of various objects such as .mp4 video format, .gif image format and other geometry formats. The XMT also supports interoperability about XML-based other players such as VRML player or SMIL player. We contain a simple converter in the content analyzer for experiment. This converter [7] is a description conversion tool that can convert scene description from the XMT to other media description languages such as SMIL or VRML.

Figure 5 shows scene description format change processing through the converter and two application examples.

The XMT of MPEG-4 content is changed into VRML or SMIL description through the converter. And, each description is separately played to the VRML player and SMIL player. For SMIL description experiment, we used RealOne player. For VRML description experiment, we used Cortona player.

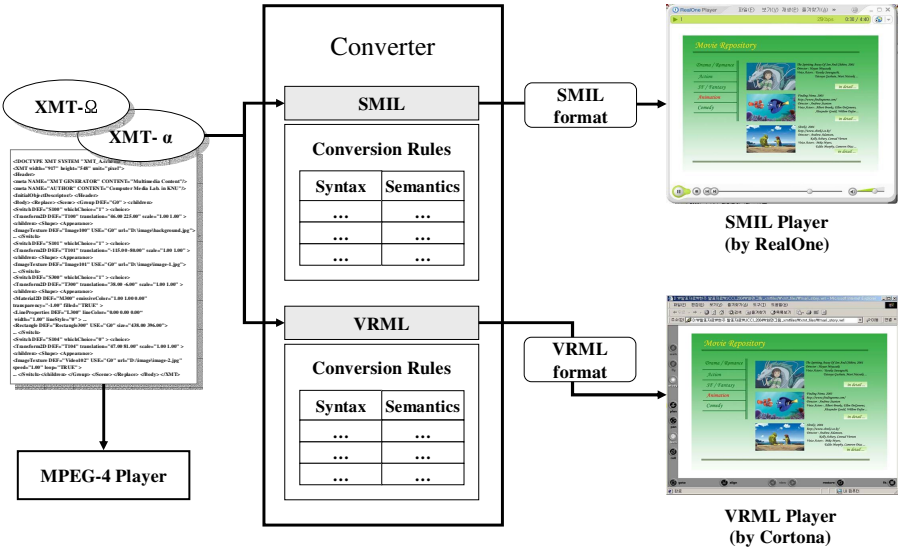


Fig. 5. The scene description format change and two application examples

The XMT format can be played in each player as well as the MPEG-4 player through language conversion. The converter changes XMT file to each description according to conversion rules of SMIL (or VRML). Conversion rule should be defined according to SMIL (or VRML) syntax and semantics. Two converted description formats can be played to each player.

The XMT can change more easily scene description's contents and attributes in user's point of view. If user knows basic syntax of XML, a user can easily change MPEG-4 content's attribute after calling various editors.

3.2 Evaluation Results

To evaluate the content analyzer's capability, the content analyzer is embedded in KNU player and we compared our content analyzer in KNU player with IM1-2D player (ver 5.0). Two players are based on MPEG-4 standard which can play the MPEG-4 content. Specially, the IM1-2D player is an open architecture player. The core comparison item is the number of searches for each tree for comparison of effective scene information management. The content analyzer contains the search engine for effective tree search and the IM1-2D player also contains search module.

The number of nodes of the MPEG-4 content as follows:

O_n: The number of objects which is defined in MPEG-4 content.
C_n: The number of conditional nodes
I_n: The number of interpolator nodes (including Color and Position2D)
R_n: The number of Routes nodes
AT_n: The number of time nodes
S_n: The sum of search of object information

$$S_{n1} = O_n + C_n + (R_n \times 2) + (AT_n \times 2) \quad (1)$$

$$S_{n2} = O_n + C_n \quad (2)$$

S_{n1} is a sum of the number of searches for all object information which contains in MPEG-4 content. It contains the number of objects, the number of conditional nodes, the number of routes nodes and the number of time nodes. Routes information is searched two times for information search because it consists of source object and target object. Time information is also searched two times for its processing because it consists of start time and end time individually.

In S_{n2}, additional tree search for routes and time information is unnecessary because all information which needs to rendering and event processing is stored in the information manager. If information manager is created, it can reduce frequent tree search that is performed whenever scene changes.

Table 1 shows several performance capability of the proposed content analyzer.

Table 1. The support capability of the content analyzer

Requirement	Support Item
Support Objects	Video(MPEG-1, MPEG-4 Video(.mp4), H.263) Audio(MPEG-4 AAC(mono, stereo), G.723) Image(BMP, JPEG, GIF format) 2D Geometry (Rectangle, Circle, Line) Text(English and Korean)
Spatial-temporal interaction	Fully supported
Independent management of each module	Fully Supported
Spatial-temporal composition	2D profiles (simple and complex scenes)
Independence of description	Fully supported
Integration with other players	Supported (with VRML player or SMIL player)
Interoperability of content	Supported (by the Converter)

4 Conclusions

In this paper, we proposed the content analyzer supporting interoperability of MPEG-4 content in heterogeneous players that is optimized for multiple scene description and management of the scene information. It consists of the description separator, the BIFS/XMT engine and the information manager. For multiple scene description support and extensibility of MPEG-4 content, we defined the description separator and the BIFS/XMT engine. For scene information management, we defined the informa-

tion manager. The information manager improves reusability and flexibility by object-based that is MPEG-4 scene's characteristic, and it also reduces the number of tree search. In order to expand module easily and improve player's throughput, each module in content analyzer is independently managed.

In the future, to support sufficiently spatial-temporal relationships of content, we will improve the content analyzer's composition capability, various events processing and playback quality in MPEG-4 player that requires high QoS.

References

1. R. Koenen, "MPEG-4 Overview," ISO/IEC JTC/SC29/WG11/N4668, March 2002, http://www.m4if.org/resources/Overview.pdf
2. A. Puri and T. Chen, Multimedia Systems, Standards, and Networks, Marcel Dekker, New York, USA, March 2000
3. M. Kim, M. Bourges-Sevenier, J. Dufourd, R. Rafey, S. Wood, L. Herrmann, Y. Fisher, Z. Lifshitz and C. Herpel, "Study Text of WD2.0 ISO/IEC 14496-1:2002 – Chapters 14-17," ISO/IEC JTC/SC29/WG11/N4848, May 2002
4. A.E. Walsh and M. Bourges-Sevenier, MPEG-4 Jump-Start, Prentice Hall, Upper Saddle River, NJ, USA, December 2001
5. K. David, K. Padhi, S. George, "Implementation of MPEG 4 systems and delivery layer," Information Technology: Research and Education, Proceedings of the ITRE2003, pp. 27-31, August 2003
6. H. Lee, S. Kim, ""An Adaptive Scene Compositor Model in MPEG-4 Player for Mobile Device" LNCS 3331: Proceedings of 5th pacific Rim Conference on Multimedia, pp. 461-469, Tokyo Japan, November 2004
7. H. Lee, Y. Lim and S. Kim, "An MPEG-4 System supporting XMT Scene Description," Proceedings of the 14th Joint Conference on Communications and Information, pp. II-D-2.1-4, Chungmu, Korea, April 2004

Adaptive Voice Smoother with Optimal Playback Delay for New Generation VoIP Services

Shyh-Fang Huang[1], Eric Hsiao-Kuang Wu[2], and Pao-Chi Chang[3]

[1] Department of Electrical Engineering, National Central University, Taiwan
hsf@vaplab.ee.ncu.edu.tw
[2] Department of Computer Science & Information Engineering, National Central University, Taiwan
hsiao@csie.ncu.edu.tw
[3] Department of Communications Engineering, National Central University, Taiwan
pcchang@ee.ncu.edu.tw

Abstract. Perceived voice quality is a key metric in VoIP applications. The quality is mainly affected by IP network impairments such as delay, jitter and packet loss. Playout buffer at the receiving end can be used to compensate for the effects of jitter based on a tradeoff between delay and loss. Adaptive smoothing algorithms are capable of adjusting dynamically the smoothing time based on the network parameters to improve voice quality. In this article, we introduce an efficient and easy perceived quality method for buffer optimization to archive the best voice quality. This work formulates an online loss model which incorporates buffer sizes and applies the Lagrange multiplier approach to optimize the delay-loss problem. Distinct from the other optimal smoothers, the proposed optimal smoother is suitable for any codec and carries the lowest complexity. Simulation experiments validate that the proposed adaptive smoother archives significant improvement in the voice quality.

1 Introduction

The rapid progress of the development of IP-base network has enabled numerous applications that deliver not only traditional data but also multimedia information in real time. The next generation network, like an ALL-IP network, is a future trend to integrate all heterogeneous wired and wireless networks and provide seamless world-wide mobility. In an All-IP network, one revolution of the new generation Internet applications will realize VoIP services that people can talk freely around through the mobile-phones, the desktops and VoIP telephones at any time and place. Unfortunately, the IP-based networks do not guarantee the available bandwidth and assure the constant delay jitters (i.e., the delay variance) for real time applications. In other words, individual transmission delays for a given flow of packets in a network may be continuing to change caused by varying traffic load and differing routing paths due to congestions, so that the packet network delays for a continuous series of intervals (i.e. talkspursts) at the receiver may not be the same (i.e. constant) as the sender. In addi-

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 1006 – 1015, 2005.
© IFIP International Federation for Information Processing 2005

tion, a packet delay may introduce by the signal hand-out or the difference of bandwidth transportation in wireless/fixed networks.

For delay sensitive applications, a dominating portion of packet losses might be likely due to delay constraint. A late packet that arrives after a delay threshold determined by playback time is treated as a lost packet. A tight delay threshold not only degrades the quality of playback but also reduces the effective bandwidth because a large fraction of delivered packets are dropped. In fact, delay and loss are normally not independent of each other. In order to reduce the loss impact, a number of applications utilize an adaptive smoothing technique in which buffers are adopted to reduce the quality damage caused by loss packets. However, a large buffer will introduce excessive end-to-end delay and deteriorate the multimedia quality in interactive real-time applications. Therefore, a tradeoff is required between increased packet loss and buffer delay to achieve satisfactory results for playout buffer algorithms.

In the past, the works on the degradation of the voice quality consider the effect of packet loss, but not that of packet delay. Within literature on predicting delays, the use of Pareto distribution in [1] is of computing the distribution parameters and rebuilding the new distribution to predict the next packet delay, and the use of neural network models to learn traffic behaviors [2]. The use of Pareto distribution or a neural network model requires relatively high complexity or a long learning period. Therefore, we consider the smoothers [3]-[9] which employ statistical network parameters related with the voice characteristic, i.e. loss, delay and talk-spurt that have significant influence to the voice quality. They detect delay spike in traffic and quickly calculate the required buffer size to keep the quality as good as possible.

For perceptual-based buffer optimization schemes for VoIP, voice quality is used as the key metric because it provides a direct link to user perceived QoS. However, it requires an efficient and accurate objective way to optimize perceived voice quality. In this paper, we propose a new delay-loss smoother that employs the Lagrange multiplier method to optimize the voice quality by balancing the delay and the loss. Lagrange multiplier method is often used to optimize the trade off problems. The contributions of this paper are two-fold: (i) A new method is for optimizing voice quality for VoIP and is easily applied to new codecs. (ii) Different from the other optimal smoothers, our optimal smoother has the lowest complexity with $O(n)$. The remainder of the paper is structured as follows. Section 2 reviews the previous work. Section 3 proposes the novel adaptive smoother. Section 4 shows the simulation results in smoothers. Finally, conclusions are provided.

2 Related Work

The SD algorithm has been studied by many researchers [3]-[9]. A delay spike is defined as a sudden and significant increase of network delay in a short period often less than one round-trip. This algorithm adjusts the smoothing size, i.e. playback delay, at the beginning of each talk-spurt. The results of this algorithm are therefore compared to the results obtained herein.

The SD Algorithm in [3] estimates the playout time p_i of the first packet in a talk-spurt from the mean network delay d_i and the variance v_i for packet i as

$$p_i = t_i + d_i + \gamma v_i \tag{1}$$

where t_i represents the time at which packet i is generated at the sending host and γ is a constant factor used to set the playout time to be "far enough" beyond the delay estimate such that only a small fraction of the arriving packets could be lost due to late arrival . The value of $\gamma = 4$ is used in simulations [3]. The estimates are re-computed each time a packet arrives, but only applied when a new talk-spurt is initiated.

The mean network delay d_i and variance v_i are calculated based on a linear recursive filter characterized by the factors α and β.

$$\begin{cases} If\ n_i > d_{i-1} \Rightarrow \begin{cases} d_i = \beta d_{i-1} + (1-\beta) n_i \\ v_i = \beta v_{i-1} + (1-\beta)\left|d_{i-1} - n_i\right| \end{cases} & (SPIKE_MODE) \\ If\ n_i \le d_{i-1} \Rightarrow \begin{cases} d_i = \alpha d_{i-1} + (1-\alpha) n_i \\ v_i = \alpha v_{i-1} + (1-\alpha)\left|d_{i-1} - n_i\right| \end{cases} \end{cases} \tag{2}$$

where n_i is the end-to-end delay introduced by the network and typical values of α and β are 0.998002 and 0.75 [3], respectively.

The decision to select α or β is based on the current delay condition. The condition $n_i > d_{i-1}$ represents network congestion (*SPIKE_MODE*) and the weight β is used to emphasize the current network delay. On the other hand, $n_i \le d_{i-1}$ represents network traffic is stable, and α is used to emphasize the long-term average.

In estimating the delay and variance, the SD Algorithm uses only two values α and β that are simple but may not be adequate, particularly when the traffic is unstable. For example, an under-estimated problem is when a network becomes spiked, but the delay n_i is just below the d_{i-1}, the SD Algorithm will judge the network to be stable and will not enter the *SIPKE_MODE*.

3 Optimal Smoother with Delay-Loss Trade Off

The proposed optimal smoother is derived using the Lagrangian method to trade off the delay and loss. This method involves, first, building the traffic delay model and the loss model. Second, a Lagrangian cost function Q is defined using this delay and the loss models. Third, the Lagrangian cost function Q is minimized and thus the delay and loss optimized solution is obtained.

3.1 Traffic Delay and Loss Models

For perceived buffer design, it is critical to understand the delay distribution modeling as it is directly related to buffer loss. The characteristics of packet transmission delay over Internet can be represented by statistical models which follow Exponential distribution for Internet packets (for a UDP traffic) has been shown to consistent with an Exponential distribution [10]. In order to derive an online loss model, the packet end-to-end delay is assumed as an exponential distribution with parameter $1/\mu$ at the receiving end for low complexity and easy implementation. The CDF of the delay distribution $F(t)$ can also be represented by [11]

$$F(t)=1-e^{t\mu^{-1}} \tag{3}$$

and the PDF of the delay distribution $f(t)$ is

$$f(t)=\frac{dF(t)}{dt}=\mu^{-1}e^{-t\mu^{-1}} \tag{4}$$

In a real-time application, a packet loss that is solely caused by extra delay can be derived from the delay model $f(t)$. Figure 1 plots the delay function $f(t)$, which shows that when the packet delay exceeds the smoothing time; the delayed packet is regarded as a lost packet. The loss function $l(b)$ can be derived from Fig. 1 as

$$l(b)=\int_b^{\infty}f(t)dt=\left(-e^{-\mu^{-1}t}\right)\Big|_b^{\infty}=-e^{-\infty}+e^{-\mu^{-1}b}=e^{-\mu^{-1}b} \tag{5}$$

From Eqs. (4) and (5), we obtain the delay and loss functions that will be used in Lagrangian cost function.

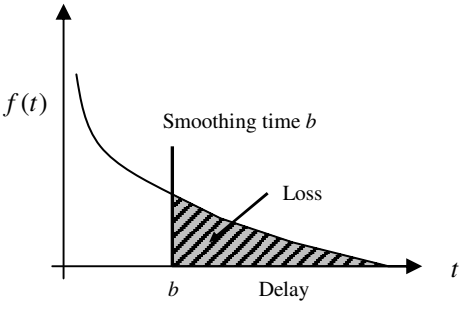


Fig. 1. The relation of smoothing delay and loss

3.2 Optimal Delay-Loss Adaptive Smoother

To express the corresponding quality for a given voice connection, a Lagrangian cost function Q is defined based on the delay b and the loss model $l(b)$

$$Q(b)=b+K\cdot l(b) \tag{6}$$

where $Q(b)$ represents the negative effect on voice quality, i.e., minimizing Q yields the best voice quality. K is a Lagrange multiplier where the loss becomes more significant as K increases. The K value has significant influence on the optimization process. We will discuss the valid range of the value in this section and the suggested value in the next section.

Here, once a smoothing time b is specified, the loss $l(b) = e^{-\mu^{-1}b}$ can be calculated from Eq. (5). The Lagrangian cost function in Eq. (6) yields

$$Q(b) = b + K \cdot e^{-\mu^{-1}b} \tag{7}$$

The differential equation dQ/db is assigned to zero that minimizes Q to yield the smoothing time b,

$$b = \mu \ln\left(K\mu^{-1}\right) \tag{8}$$

where b is the best smoothing time for balancing the delay and the loss. Afterwards, the smoother can provide best quality, considering both the delay and the loss effects, based on the calculated smoothing time b.

The calculated smoothing time b is a function of K and μ. μ denotes a IP-base network delay parameter (end-to-end delay) and can be measured at the receiver, but K is given by users or applications. The calculated smoothing time b must be within an allowable range to ensure that the end-to-end delay is acceptable. Here, D_{max} is defined as the maximum acceptable end-to-end delay and the calculated smoothing time b must be between 0 and D_{max}

$$0 \le b = \mu \ln\left(K\mu^{-1}\right) \le D_{max} \tag{9}$$

Accordingly, the permissible range of valid K in the Lagrange multiplier Q function in Eq. (8) is

$$\mu \le K \le e^{D_{\max} * \mu^{-1}} * \mu \tag{10}$$

3.3 Suggestion of *K* Parameter

In this section the relationship between the voice quality and loss is further studied. Based on the previous section discussions, we know K parameter is tightly related with voice quality. In other words, for a given MOS (Mean Opinion Score) of speech quality, the allowable range of K can further be restricted. Many studies revealed the difficulty of determining the mathematical formula that relates the voice quality, delay, and loss. According to [12], the loss degrades the voice quality more remarkably than does the delay, so the quality-loss relationship is first emphasized [13] [14]. In these studies, an empirical Eq. (11) was obtained by experiments with many traffic

patterns for predicting the voice MOS quality MOS_{pred} that might be degraded by the traffic loss ($loss$)

$$MOS_{pred} = MOS_{opt} - c * ln(loss + 1) \tag{11}$$

where MOS_{opt} is voice codec related, representing the optimum voice quality that the codec can achieve, c is a constant that is codec dependent, and $loss$ is a percentage ratio times 100. Following this approach, anyone can estimate a specific empirical rule with specified voice codecs and network environments. Equation (11) also implies that the network loss rate must be kept lower than or equal to the defined $loss$ to ensure the predicted MOS MOS_{pred} .

Equation (11) is rewritten to yield Eq. (12),

$$loss = 2^{\frac{MOS_{opt} - MOS_{pred}}{c}} - 1 \tag{12}$$

Notably, the $l(t)$ function is a percentage but loss is not. Therefore, $l(t)$ is multiplied by 100 to yield

$$loss = 2^{\frac{MOS_{opt} - MOS_{pred}}{c}} - 1 \geq l(t) = e^{-\mu^{-1} b} * 100 \geq 0 \tag{13}$$

From Eq. (13), the smoothing time b is

$$b \geq -ln\left(\frac{2^{\frac{MOS_{opt} - MOS_{pred}}{c}} - 1}{100} \right) * \mu \tag{14}$$

From Eqs. (8), (10) and (14), the suggested range for K is

$$K \geq max\left(\frac{100 * \mu}{(2^{\frac{MOS_{opt} - MOS_{pred}}{c}} - 1)}, \mu \right) \tag{15}$$

When K is assigned a value that is more than the threshold in Eq. (15), the design of the smoother is mainly dominated by the loss effect. For a given MOS, a suitable K can be suggested and an optimal buffer size can be determined.

4 Simulation

4.1 Simulation Configuration

A set of simulation experiments are performed to evaluate the effectiveness of the proposed adaptive smoothing scheme. The OPNET simulation tools are adopted to

trace the voice traffic transported between two different LANs for a VoIP environment. Ninety personal computers with G.729 traffics are deployed in each LAN. The duration and frequency of the connection time of the personal computers follow Exponential distributions. Ten five-minute simulations were run to probe the backbone network delay patterns, which were used to trace the adaptive smoothers and compare the effects of the original with the adapted voice quality latter.

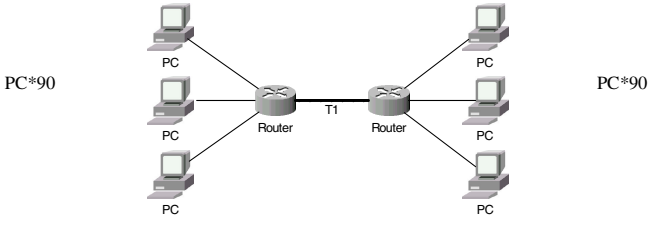
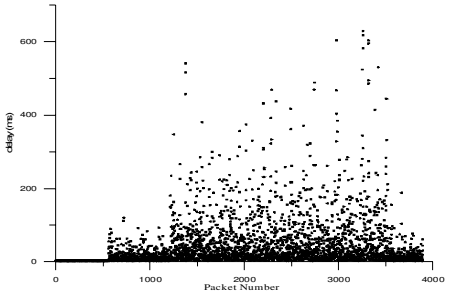


Fig. 2. The simulation environment of VoIP

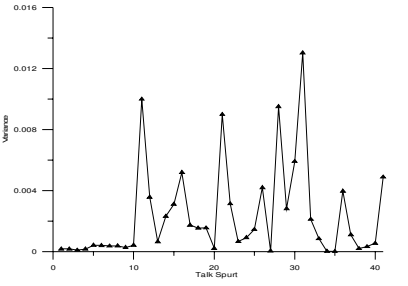
Table 1. Simulation parameters

Attribute	Value
Numbers of PC in one LAN	90 PCs
Codec	G.729
Backbone	T1 (1.544 Mps)
LAN	100 Mbps
Propagation delay	Constant
Router buffer	Infinite
Packet size	50 bytes

Fig. 2 shows the typical network topology in which a T1 (1.544 Mbps) backbone connects two LANs, and 100 Mbps lines are connected within each LAN. The propagation delay of all links is assumed to be a constant value and will be ignored (the derivative value will be zero) in the optimization process. The buffer size of the bottlenecked router is assumed to be infinite since the performance comparison of adaptive smoothers will be affected by overdue packet loss (over the deadline) and not affected by the packet loss in router buffer. The network end-to-end delay of a G.729 packet with data frame size (10 bytes) and RTP/UDP/IP headers (40 bytes) is measured for ten five-minute simulations by employing the OPNET simulation network. Table 1 summarizes the simulation parameters. Figure 3(a) and 3(b) list one of the end-to-end traffic delay patterns and the corresponding delay variances for VoIP traffic observed at a given receiver.



(a) The delay of traffic



(b) The variance of traffic

Fig. 3. VoIP traffic pattern

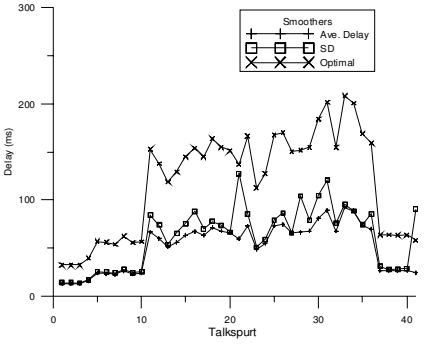


Fig. 4. The predicting time of smoothers

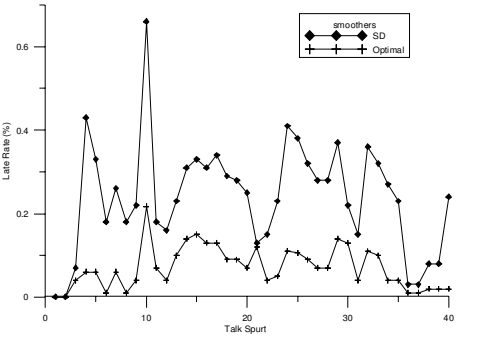


Fig. 5. The packet loss rate of smoothers

4.2 Predicted Smoothing Time and Loss Rate in Smoothers

In this section the accuracy of the predicted end-to-end delay time and loss rate among these smoothers are compared. The maximum acceptable delay D_{max} is set to 250 ms and the average delay is used to observe the traffic pattern in particular. In Fig. 4 and Fig. 5, we can observe that the predicted time of the SD smoother is very close to the mean delay and the loss rate is higher than optimal smoother. The SD smoother uses a large value of fixed β to deal with various traffic conditions and emphasize a long-term mean delay d_{i-1}, so the predicted delay will be close to the mean delay. A better choice for n_i is probably the maximum delay in the last talk-spurt that can sufficiently represent the worst case of current network congestion and avoid an under-estimated delay.

4.3 Quality Measurement

The test sequence is sampled at 8 kHz, 23.44 seconds long, and includes English and Mandarin sentences spoken by male and female. Table 2 lists the mean delay, mean

loss rate, and SSNR measured in a voice quality test with various smoothers. SSNR [15][16] is used as an evaluation tool because it correlates better with MOS and it is relatively simple to compute. Table 2 shows that the Optimal smoother performance achieves a high average SSNR and has the significant improvement in the voice quality over SD smoother, since the proposed optimal smoother truly optimizes with the delay and loss impairments.

The SSNR can only represent the loss impact, but hardly represent the delay impact. Therefore, a Lagrangian cost function is utilized to consider the delay and loss impacts to the quality degradation for various smoothers. In order to maintain the normal voice quality over the network, the predicted MOS, $MOS_{pred} = 3$ is required. According to [14] and G.729, c is set as 0.25 in formula (15) and the μ is set as the frame rate 10 ms for G.729 at the sender. The Lagrange multiplier value $K = 430$ is calculated from the formula (15). Table 3 shows that the optimal smoother has the lower Lagrangian cost value than SD smoother. Specifically, we can observe the optimal smoother has 23% improvement of the quality degradation on SD smoother.

Table 2. The voice quality test of smoothers

	Source	SD	Optimal
SSNR (dB)	8.17	5.67	7.51
Mean delay (ms)		89.22	112.46
Mean Loss Rate (%)		0.21	0.09

Table 3. The mean negative cost value of smoothers at high traffic load

Smoother	SD	Optimal
Lagrangian Cost (ms)	220.2166	170.2838

5 Conclusion

For new-generation VoIP services, a dynamic smoothing algorithm is required to address IP-based network delay and loss. This work proposes an optimal smoothing method to obtain the best voice quality by Lagrangian lost function which is a trade off between the negative effects of the delay and the loss. It can efficiently solve the mismatch between the capture and the playback clocks. Numerical examples have shown that our proposed method can control the playout time to balance the target delay and loss.

References

1. Brazauskas V., Serfling R.: Robust and efficient estimation of the tail index of a one-parameter pareto distribution. North American Actuarial Journal available at http://www.utdallas.edu/~serfling. (2000)

2. Tien P. L., Yuang M. C.: Intelligent voice smoother for silence-suppressed voice over internet. IEEE JSAC, Vol. 17, No. 1. (1999) 29-41
3. Ramjee R., Kurise J., Towsley D., Schulzrinne H.: Adaptive playout mechanisms for packetized audio applications in wide-area networks. Proc. IEEE INFOCOM. (1994) 680-686
4. Jeske D. R., Matragi W., Samadi B.: Adaptive play-out algorithms for voice packets. Proc. IEEE Conf. on Commun., Vol. 3. (2001) 775-779
5. Pinto J., Christensen K. J.: An algorithm for playout of packet voice based on adaptive adjustment of talkspurt silence periods. Proc. IEEE Conf. on Local Computer Networks. (1999) 224-231
6. Liang Y. J., Farber N., Girod B.,: Adaptive playout scheduling using time-scale modification in packet voice communications. Proc. IEEE Conf. on Acoustics, Speech, and Signal Processing, Vol. 3. (2001) 1445-1448
7. Kansal A., Karandikar A.: Adaptive delay estimation for low jitter audio over Internet. IEEE GLOBECOM, Vol. 4. (2001) 2591-2595
8. Anandakumar A. K., McCree A., Paksoy E.: An adaptive voice playout method for VOP applications. IEEE GLOBECOM, Vol. 3. (2001) 1637-1640
9. DeLeon P., Sreenan C. J.: An Adaptive predictor for media playout buffering. Proc. IEEE Conf. on Acoustics, Speech, and Signal Processing, Vol. 6. (1999) 3097-3100
10. Bolot J. C.: Characterizing end-to-end packet delay and loss in the internet. Journal of High-Speed Networks, Vol. 2. (1993) 305-323
11. Huebner F., Liu D., Fernandez J. M.: Queueing Performance Comparsion of Traffic Models for Internet Traffic. GLOBECOM 98, Vol. 1. (1998) 471–476
12. Nobuhiko K. and Kenzo I.: Pure delay effects on speech quality in telecommunications. IEEE JSAC, Vol. 9, No. 4. (1991)
13. Duysburgh B., Vanhastel S., De Vreese B., Petrisor C., and Demeester P.: On the influence of best-effort network conditions on the perceived speech quality of VoIP connections. Proc. Computer Communications and Networks. (2001) 334-339
14. Yamamoto L., Beerends J., KPN Research.L: Impact of network performance parameters on the end-to-end perceived quality. EXPERT ATM Traffic Symposium available at http://www.run.montefiore.ulg.ac.be/~yamamoto/ publications.html. (1997)
15. Melsa P. J. W., Younce R. C., and Rohrs C. E.: Joint impulse response shortening for discrete multitone transceivers. IEEE Trans..Communications, Vol. 44, No. 12. (1996) 1662-1672
16. Hosny N. M., El-Ramly S. H., El-Said M. H.: Novel techniques for speech compression using wavelet transform. The International Conference on Microelectronics. (1999) 225-229

Designing a Context-Aware System to Detect Dangerous Situations in School Routes for Kids Outdoor Safety Care

Katsuhiro Takata[1], Yusuke Shina, Hiraku Komuro, Masataka Tanaka, Masanobu Ide, and Jianhua Ma

[1] Graduate School of Computer and Information Sciences, Hosei University, Tokyo, Japan
i04t9002@cis.k.hosei.ac.jp
Faculty of Computer and Information Sciences, Hosei University, Tokyo, Japan
{shina, komuro, tanaka, ide}@malab.k.hosei.ac.jp
jianhua@k.hosei.ac.jp

Abstract. Ubiquitous computing is targeted at services and applications of computer and communication technologies in the real world. This research, as a part of UbicKids Project, is focused on designing a context-aware system that dynamically detects the possible dangerous situations in the routes where kids go to and return from schools, and provides prompt advices to kids who may encounter some dangerous situations. Based on analyses of typical dangerous situations in school routes, the paper then shows the system architecture and discusses about danger-related context information processing including context description, representation and presentation. Security and privacy issues and possible solutions are also explained. A preliminary system prototype has been implemented and some GUIs are explained. Related work is discussed with comparisons to other research work.

1 Introduction

Since Weiser's pioneering work, it has been recognized that communications between small embedded sensors and related data processing devices are an integral facility in many ubiquitous computing scenarios. It is common that context-aware services [1-3] are involved in a huge amount of spatial and other contextual information to be sensed, exchanged and processed among a number of pervasive devices. Such ubiquitous computing and communication have being opened a great many of opportunities to provide novel solutions to various issues in the real human life.

Caring children is one kind of common human activities, and consumes a lot of time/energy to many parents. It is a fact that parents cannot always watch their kids and give them prompt supervisions/helps in 24/7, but they do expect their kids to be well taken care of with their preferred means in every place at all times. Therefore, we started the Project UbicKids [4], a smart hyperspace environment of ubiquitous care for kids, from early 2004. There are lots of kids caring activities to be supported, which can be basically divided into three categories: kids awareness, kids assistance and kids advice, i.e., 3As. Among these, a ubiquitous service strongly desired by parents is to have some system that can help them to take care of kids safety, especially

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 1016 – 1025, 2005.
© IFIP International Federation for Information Processing 2005

when the kids are out of home and at somewhere outside. One popular case is when primary school kids are on the way to school/home by themselves.

Thus, this research is focusing on designing a ubiquitous kids safety care system to dynamically detect possible dangerous situations in school routes and promptly give advices to kids and/or their parents in order to avoid or prevent from some possible dangers. To detect the dangerous situations, it is essential to get enough contexts of real environments in kids surroundings. This is based on two basic assumptions: (1) a big number of sensors, RFIDs, tags and other information acquisition devices are pervasively distributed somewhere in and near school routes, and (2) a kids should carry or wear some devices that can get surrounding context data from the above pervasive devices. The data amount received by the kids devices are often enormous and dynamically changed during their walks on the roads. One of core issues is how to effectively process the enormous and varied context information for the system to automatically and correctly recognize what and where dangers will be encountered by the kids so as to take the further actions to avoid them by advising the kids. Such complex context processing is currently impossible to fully complete with only using devices carried by kids since their computing performance is relatively low. One solution is that the kids devices only make simple processing on the sensed contexts and then send them to some powerful host to have further processing. Due to some problems including security [5, 6], the host, called safety server, should be at a kid's own home. The safety server will not only process contexts but also analyze changed situations, find possible dangers, advise kids, etc. It also plays a role of mediators to communicate between kids, parents and other public information services. This paper presents our preliminary research on such a safety care system, which seems the first in the world.

In the rest of the paper we first describe computing scenarios of the kids safety care services in the next section, and discuss our system architecture including context descriptions and processing flows in Section 3. Next, Section 4 explains our considerations about possible solutions to the security and privacy issue. Section 5 shows how the advice information is presented on some prototype GUIs to kids and parents. Related work is given in section 6, and conclusions are addressed in the last section.

2 Computing Scenarios of Kids Safety Care in School Routes

Figure 1 shows a system overview of computing scenarios to support kids safety care in their school routes. The whole system can be described with five basic entities: school kid, kids surrounding, remote parent, information provider, and safety server. A school child should carry some devices that dynamically collect context data from his or her surrounding. The surrounding context data is sent to and processed by a corresponding safety server located at the kid's home. The safety sever may need to get the further necessary surrounding information from some related information provider. When needed, the safety server will send situational information around a kid to his/her parent, and act as a mediator between the kid and the parent.

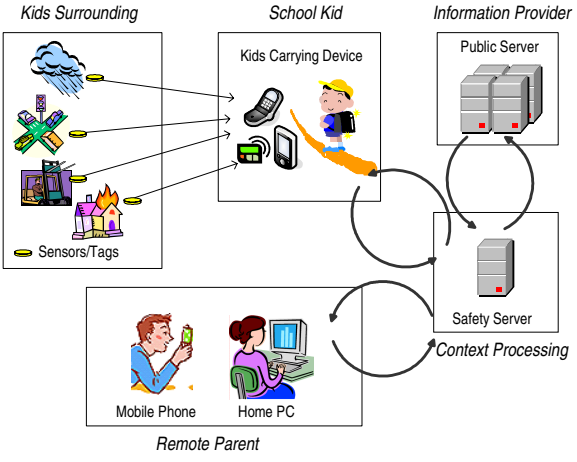


Fig. 1. System use scenarios and process relationships among the machines and devices in outdoor, home, information provider and the users

Theoretically, a kid's device fully satisfying the safety care should have the following general features: (1) realtime acquiring the necessary kids surrounding context from the sensing and tagged devices distributed on the real world, (2) always knowing its current position information, (3) fast communicating with the safety server, (4) necessary multimedia interface for presenting easily noticeable advices, (5) reasonable continuous working period or low power consumption, and (6) compact, light, reliable, etc. Such a device is greatly useful not only for kids safety care but also other care functions, and even in many other ubiquitous applications. It does not exist yet at the present and would be available in near future. However, it is possible to use current available devices, such PDA, cell phone, compact game machine and some wearable devices, as some substitutes to build some prototype system at the current stage. Although it takes time to really deploy a number of the sensing and tagged devices to the real physical environments, many types of these devices are available, such as GPS, RFID, tiny microphone/camera, various sensors, etc. They can be used in making prototype systems and conducting related experiments. The devices used by parents may be PC, PDA, cell phone, or some handhelds.

The safety server can run on an ordinary PC. A tag like RFID may not include enough information about a physical state but can indicate where to get its more information. The detailed information about a dangerous site, such as a road cross, traffic accident, fire event, or others, may be put some public server. For security considerations, the kid devices will not directly access the public server, and the safety server, after getting the information address, e.g., URL/URI, accesses the public server to get further necessary information. Generally speaking, the safety server should not only continually track a kid's movements and acquire its surrounding contexts, but also know the situated contexts about parents and their surrounding so as to actively or proactively send the kid related urgent information to the parents in right place, suitable time and proper manner.

3 System Requirements and Architecture

The main objective in designing this system is to meet the following requirements:

- *Context data collection*. The system should have a data collection service of spatial information related to a kid and his/her surroundings. This service provides the space related information to the safety server.
- *Dangerous situation detection*. The system should facilitate detecting possible dangers with using the space information and other contexts. The detection service must be based on some semantic model of dangerous situations and general data processing schemas.
- *Device dependent presentation.* The information should be properly presented with adapting to various devices. The most commonly, kid's devices (such as PDA, mobile phone and other similar handheld devices) may have less function of presentation since the dust devices have limited presentation power. On the other hand a PC used by parent is often with rich representation power.

In designing the safety care system, these fundamental requirements are essential to build rational and feasible system architecture. In order to detect a possible danger somewhere around a kid, and inform the kid and his/her parents of it, the architecture should interconnect four main roles: kid, parent, safety server and data store. When a kid is on the way to school, a kid's device scans various data such as whether, building, traffic and so on. The surrounding data may contain sufficient information related to the state of a physical event, or partial information only with a state ID code/string, such as a URL, from which the safety server can get more detailed information about the state via accessing a corresponding database or data store offered by an associated information provider. With using all available information, the safety server analyzes possible hidden dangers, and, when necessary, sends some alerts to a kid and/or parents.

Usually, a large amount of surrounding data in raw data types is dynamically acquired during a kid's way to home/school. To smoothly perform efficient operations, the data must be semantically represented with some metadata [7-9] and the data amount has to be greatly reduced by abstracting only needful information to describe meaningful dangerous events. Figure 2 shows an example of a fire related context representation.

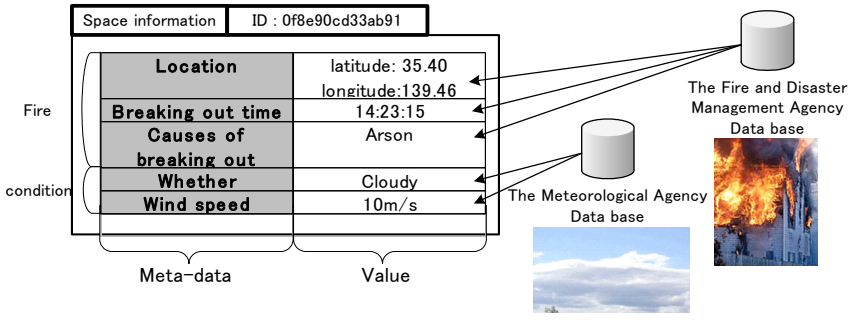


Fig. 2. Fire context information (top of table) and conditions (bottom of table) contained in space information are presented metadata (left of table) and value (left of table)

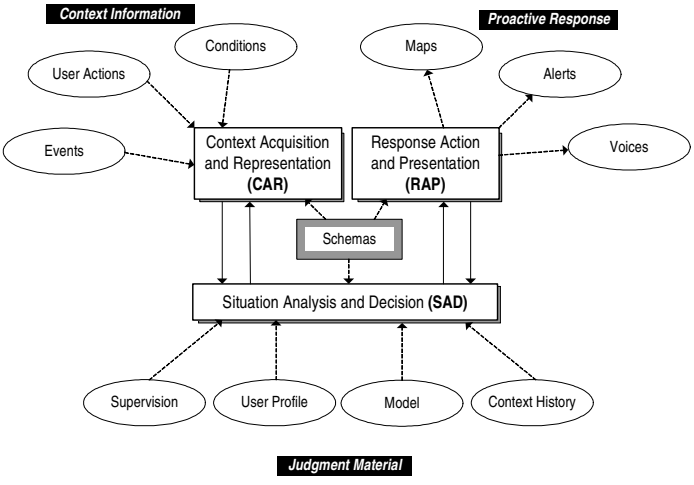


Fig. 3. Context processing functions and flows. Context processing in all modules is based on schemas since context information may be in various representations or regulations.

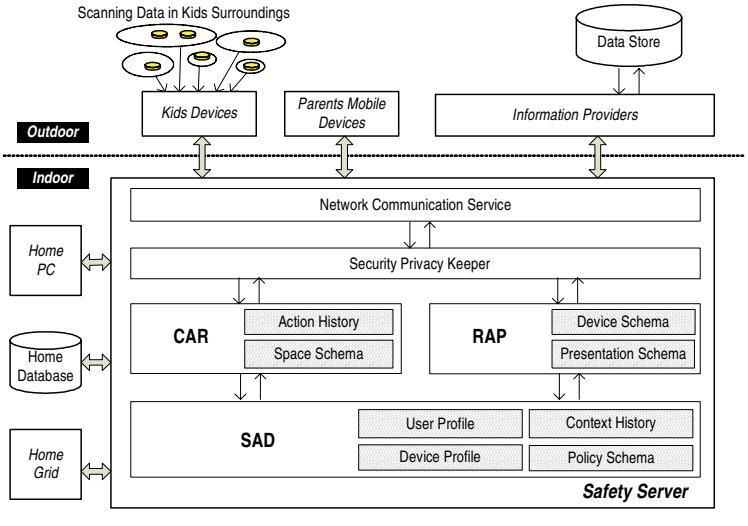


Fig. 4. System architecture consists of internal processing flows of main functions and external communications via networks

The core of the safety server is to effectively process the context data in real time, and precisely adapt to various situations in kids safety care. It, as shown in Fig. 3, consists of three basic functions: Situation-Analysis-and-Decision (SAD), Context-Actuation-and-Representation (CAR) and Response-Action-and-Presentation (RAP). And Fig. 4 gives the whole system architecture.

CAR is to fulfill two main tasks, one is to acquire the context data from the kid's devices and other sources, and the other is to represent the context data with meaningful metadata and extract the useful data so as to reduce data amount. SAD plays a central role in this architecture since it is to analyze a situation using semantic context information provided by CAR, make a judgment if the situation is or to become dangerous, and decide what action(s) should be taken to avoid the danger. When an action is decided by SAD, corresponding instructions and associated data will be given to RAP that completes the action via presenting some information to some devices used by a kid and/or parent with adapting to their available device types and other surrounding states. Besides the above three basic functions the architecture also includes other important functions: *network communication services* to mange communications between devices/machines inside and outside homes, *security privacy keeper* to guarantee security of data and communication, and *home database/grid* to further enhance data storage and processing abilities.

4 System Security and Privacy

The system is designed based on a fundamental principle that a kid device is able to directly communicate with the safety server at home but can only passively received information from surrounding devices in the outside without actively communications with other machines not at home. This will greatly reduce the possible security holes and move the major security management work to the home safety sever that takes a full responsibility to control all authentications in accessing the system. We adopt the S/KEY authentication scheme [10, 11] that protects user passwords against passive attacks. It can be easily and quickly added to almost any UNIX system, without requiring any additional hardware and the system to store information that would be more sensitive than the encrypted passwords already stored. Figure 5 shows how the S/KEY authentication scheme works.

First, when a client make a request to login to a home safety server, the sever generator produces a random number and sends it to the client. Next, the client generator creates a one-time password with using the random number, a sequence number and a hash function, and sends its one-time password [11] to the server. Then, the generator in the server also generates a one-time password and the verification compares with the two one-time passwords. If they match, a permission is sent to the client. Finally, the calculator subtracts one from the sequence number for preparation of a next connection. In the authentication with password, the key of security intensity is how to maintain a secret of the password completely. Even if we configure sophisticated password, once a malicious user knows it, he can do unauthorized access easily. But the S/KEY authentication scheme that, whenever one logs into, uses a different password, eliminates almost all of the risk from that. Additionally, the S/KEY authentication system doesn't have to send and receive a password itself, so we need not to worry about eavesdroppers. The information flowed to the network is only seeds generated by random numbers and sequence numbers subtracted one-by-one every time we correspond, and one-time password based on them.

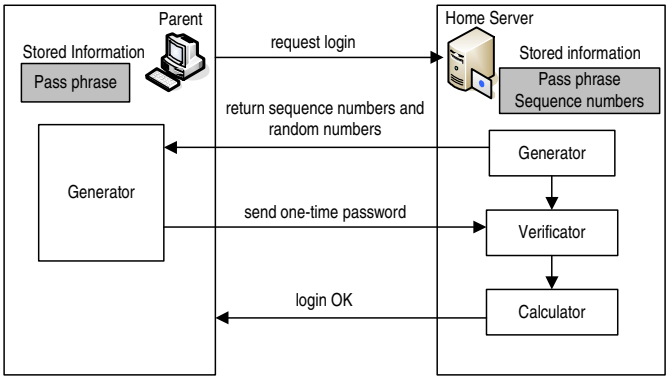


Fig. 5. The S/KEY proposal authentication includes some security modules

All users are unwilling to cope with a complex security system. There are some weaknesses in this S/KEY authentication system, but S/KEY authentication system is simple and easy to use. For this reason, it is thought that this scheme serves its purpose. Except the access authentications, communications between a safety server, kid's devices, parent's devices and data store should be also secure. Such communications can be encrypted by SSL protocol (Secure Socket Layer). In the future, constructing better security schemes and related systems will be a major challenge in solving security and privacy problems.

5 Prototype GUIs

The GUI shown in Fig. 6 is for a parent to get current kid's information. A parent runs this application on a PC, PDA or a cell phone at home or some other place whenever he/she wants to know a kid's information.

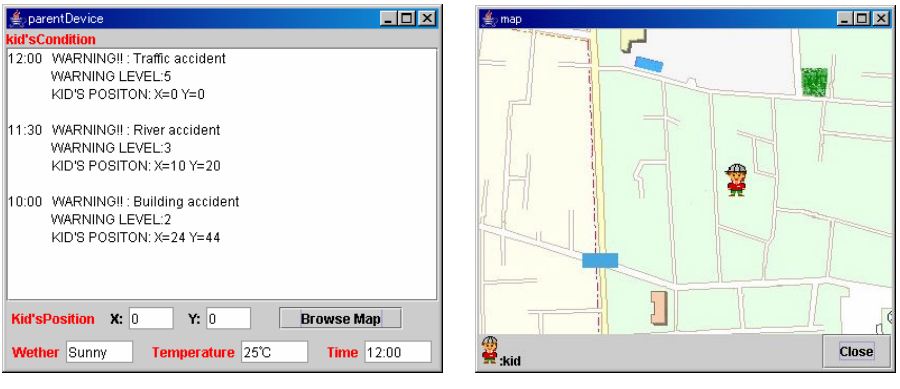


Fig. 6. Left figure is the interface on a parent's device which displays kid's dangerous information and right figure is a map showing the current location of a kid

Table 1. The information related kids space conditions in the lower pane

Kind	Form	Receive Frequency
Warning	String	When danger approaches the kids.
Warning level	Int	When danger approaches the kids.
Kid's position (x, y)	Double	Always
Weather	String	When weather change.
Temperature	Int	Always

The GUI window contains a message pane. When a parent checks a kid's situation, it will be reloaded immediately, and the related message will be shown on the message pane. In the lower part, some items are showing current kids surroundings. When warning information is coming from the safety server as a result of analysis for kids surroundings, the GUI displays the following information: the time displaying when the warning message is received, the kind of warning briefly showing what a danger event it is, the warning level illustrating a judgment of a degree of the current danger to be prevented (For the same kind of warnings, their warning levels may be different), and the kid's position where a danger event occurs, as shown both in Fig. 6 and in Table. 1.

A kid's application informs a dangerous situation to the kid immediately with a GUI shown in Fig. 7. This application can be seen as a simplified one as compared with the parent's one. This is because a kid's device may have relatively poor processing performance. Apparently noticeable information, such as short words using speech, is necessary and may be better to remind a kid a possible danger. The kid's application runs before a kid leaves home. An alarm will be given to a kid when his/her carrying device receives a warning message from the safety server. The warning level will be higher if the situation becomes more dangerous. We use the alarm sounds that are more comprehensible than displaying warning texts for effective kids notice.

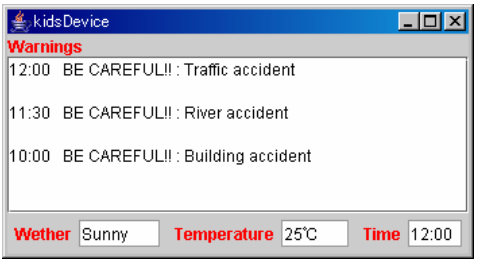


Fig. 7. Interface on a kid's device which calls a kid's attention to a dangerous situation using the alarm sounds and the warning messages

6 Related Work and Discussion

In the past, a number of systems have been developed to support context-ware computing. These systems have made progresses in various aspects of context-aware

computing but are weak in processing huge amount of context information for effectively detecting dangerous situations specifically for outdoor kids safety care. The context collecting service in our architecture can reduce the context data amount with using meta-context representations for dangerous events. MoGATU [12] is a project explicitly designed to deal with data management in the ubiquitous computing environment. A profile as well as context information is used to guide the interactions among different devices. MoGATU considers each of the devices individually to serve their users' information accesses. Their results can be used to complement our system.

Filho, et al. gives a detailed description of the design of the event notification system. There has been research in using event notifications in context-aware systems and how to notify users in a context aware manner [13]. Huang provides a good overview of the network architecture design challengers to publish or subscribe in mobile environments [14]. The technique we used in the judgment of dangerous situation is related to the decision tree algorithm. Van's work [15] under the framework of decision trees has benefited our design described in this paper.

7 Conclusion and Future Work

In this paper, we present the design of a context-aware system that dynamically detects the possible dangers in the routes where kids go to and return from schools, which is a part of our UbicKids Project [4] started from early 2004. The system architecture has been discussed in detail in terms of danger-related context information processing. Security and privacy issues and possible solutions are also explained. The semantic representation of danger-related contexts is essential topic for realizing the context-aware system. Our short-term objective is to implement the prototype system for the trial of providing kid's care and to enhance a context usage related to both users and activities by including temporal and spatial relations in ubiquitous computing environment. At present the design of the SAD and related models/methods are still in the early stage of research. Our preliminary research in the SoM based system [16] shows the space oriented diagnosis engine in our previous work. The hypothetic method based SoM looks adequate to define space information for supporting context reasoning and knowledge sharing. This method should be integrated into this architecture in near future. Of course, it is very necessary to develop a completed system prototype able to work and test so that we can find more practical problems, which can guide us to build a really useful kids safety care system. This work seems the first research aimed at building a ubiquitous system to assist the outdoor safety care of the schools kids in the real world.

References

[1] Martin Bauer, Christian Becker and Kurt Rothemel, "Location Models from the Perspective of ContextAware Applications and Mobile Ad Hoc Networks", Personal and Ubiquitous Computing, Vol. 6, No. 5, December 2002.

[2] Gregory D. Abowd. "Ubiquitous computing: Research themes and open issues from an applications perspective". Technical Report GIT-GVU 96-24, GVU Center, Georgia Institute of Technology, October 1996.
[3] Anthony Harrington, Vinny Cahill, "Route Profiling – Putting Context To Work", Proceedings of the 19th ACM symposium on Applied Computing, Nicosia, Cyprus, 2004.
[4] Jianhua Ma, Laurence T. Yang, Bernady O. Apduhan, Runhe Huang, Leonard Barolli and Makoto Takizawa, "Towards a Smart World and Ubiquitous Intelligence: A Walkthrough from Smart Things to Smart Hyperspaces and UbicKids", International Journal of Pervasive Comp. and Comm., 1(1), March 2005.
[5] Thomas Erickson, "Some Problems with the Notion of Context-aware Computing", Communications of the ACM, v.45, February 2002.
[6] Dirk Henrici, Paul Muller, "Tackling Security and Privacy Issues in Radio Frequency Identification Devices", Proceedings of Second International Conference on Pervasive Computing, Vienna, Austria, April 2004.
[7] Catriel Beeri, Tova Milo, "Schemas for Integration and Translation of Structured and Semi-Structured Data", International Conference on Database Theory, 1999.
[8] Brian Dunkey, Nandit Soparkar, "Data Organization and Access for Efficient Data Mining", Proceeding of the 15th International Conference on Data Engineering, Vienna, Austria, March 1999.
[9] Vipul Kashyap and Amit P. Sheth, "Semantic and Schematic Similarities between Database Objects: A Context-based Approach". VLDB Journal, No. 5, 1996.
[10] Liqun Chen and Chris J. Mitchell, "Comments on the S/KEY User Authentication Scheme", Operating Systems Review, 1996.
[11] Aveil D. Rubin, "Independent One-Time Passwords", Proceedings of the 5th Security Symposium, San Jose, California, July 1996.
[12] Filip Perich, Sasikanth Avancha, Dipanjan Chakraborty, Anupam Joshi, Yelena Yesha, "Profile Driven Data Management for Pervasive Environments", Proceedings 13th International Conference on Database and Expert Systems Applications, Aix-En-Provence, France, September 2002.
[13] Roberto Silveira Silva Filho, Cleidson RB de Souza, David F. Redmiles, "The Design of a Configurable, Extensible and Dynamic Notification Service", Proceedings of the 2nd International Workshop on Distributed Event-based Systems, San Diego, California, June 2003.
[14] Y. Huang and H. Garcia Molina, "Publish/Subscribe in a Mobile Environment", Proceedings of the 2nd ACM International Workshop on Data Engineering for Wireless and Mobile Access, Santa Barbara, California, May 2001.
[15] T. Van de Merckt. "Decision Trees in Numerical Attribute Spaces", Proceedings of the 13th International Joint Conference on Artificial Intelligence, Chambery, France, 1993.
[16] Katsuhiro Takata, Yusuke Shina, Jianhua Ma and Bernady O. Apduhan, "Designing a Space-oriented System for Ubiquitous Outdoor Kid's Safety Care", Proceedings of the 19th International Conference on Advanced Information Network and Applications, Taipei, March 2005.

An Advanced Mental State Transition Network and Psychological Experiments

Peilin Jiang[1,2], Hua Xiang[1], Fuji Ren[1], and Shingo Kuroiwa[1]

[1] Department of Information Science and Intelligent Systems,
Faculty of Engineering, The University of Tokushima,
Tokushima, Japan
{jiang, ren, kuroiwa, xianghua}@is.tokushima-u.ac.jp
[2] Institute of Artificial Intelligence and Robotics,
Xian Jiaotong University, Xian, China
pljiang@aiar.xjtu.edu.cn

Abstract. The study of human-computer interaction is now the most popular research domain overall computer science and psychology science. The most of essential issues recently focus on not only the information about the physical computing but also the affective computing. The emotion states of human being can dramatically affect their actions. It is important for a computer to understand what the people feel at the time. In this paper, we propose a novel method to predict the future emotion state of person depending on the current emotion state and affective factors by an advanced mental state transition network[1] . The psychological experiment with about 100 participants has been done to obtain the structure and the coefficients of the model. The test experiment also has been done to certificate the prediction validity of this model.

1 Introduction

In research of modern information science and human computer interface, the non-verbal information has caused more attention. The latest scientific research indicate that emotion of human being play a core role in decision making, perception, learning, and moreover they influence the very mechanisms of rational thinking [2]. The automatic emotion state recognition has gained more attention because of the desire to develop natural and effective interfaces for human-computer communication application [3]. The myriad of theories on emotion can be largely examined in terms of two components: emotions are cognitive, emphasizing their mental component, and emotions are physical, emphasizing their bodily component [2]. Numerous physical experiments have been done (Ekaman, Frijda, [4] [5] [6] etc.) but external information such as language and facial expressions are not enough to model human emotion [1]. Though, various emotional models have been proposed in previous studies (the Plutchik's Multidimensional Model [7], the Circumplex Model of Affect [8] etc.), there is no mental model that can be described appropriately in a numerical way.

In this paper, we present an advanced mental state transition network in order to predict the future emotion state. This network model is improved from

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 1026–1035, 2005.
© IFIP International Federation for Information Processing 2005

the original mental state transition network model[10] by considering the whole priori conditional probability under the various affective environments. The network structure and coefficients are acquired from the psychological experiment designed on the basis of the conditional propability table and be testified by 50 random data. It hypothesizes that the human emotions are simplified to seven basic categories and transfer among these discrete states. The state is defined as a mental state. However, there existed some certain expectation value with some external causes. By means of experimentation of a large set of psychological questionnaires, the conditional transition probabilities among mental states can be calculated.

The proposed advanced mental state transition network can be used in prediction of the emotion state of human on engineering science. Also the model which gathered from a large amount of raw psychological data can reflect the common aspect of human emotion transitions. The test experiment also verifies the model reliable and pragmatic.

2 Emotion State Transition Network Model

The research of human mental state firstly started on the psychological field and many definitions of emotion have been defined and psychological tradition has probed that nature of emotions systematically are shaped by major figures in several discipline-philosophy, biology and psychology [11].The two main theories in this domain have indicated that definitely some cross-culture emotions widely existed in each nationality and there exists a separation between positive and negative affective emotions [12]. Now the improvement of research on human-machine interaction makes it necessary to find out a describable definition of emotion and a practical method to recognize and predict emotions.

2.1 Prototype of Six Basic Emotions

After studying a vast amount of literature on the signs that indicate emotion, both within the psychological tradition and beyond it [3], we have found six archetypal emotions (happy, sad, angry, surprise, fear and disgust) as presented by Ekman. They are widely accepted among different areas and are easier to capture and describe than other complex emotions [11]. In our research, we presume that human mental movements can be divided into six archetypal emotional states. Besides these, we add another neutral state-calm (quiet / serene).

2.2 Mental State Transition Network

Emotion in its narrowest sense is full-blown emotion and is generally short lived and intense [13]. To simplify the study, our experiment only takes account of the full-blown emotion. Then, the seven discrete mental states we proposed can construct a closed mental space. In that case, we can create a mental state transition network model of human being as the Fig 1 shows. In the Fig 1 the

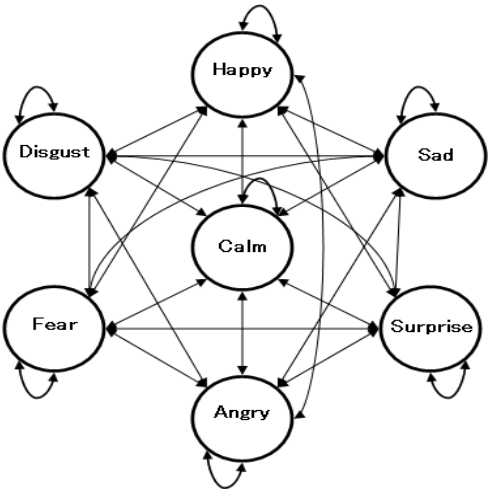


Fig. 1. Mental State Transition Network Model

circle in center indicates a calm mental state. The circles around represent the other six mental states. The arrows denote direction from one state to the other.

2.3 Improvement of the Mental State Transition Network

The original mental state transition network is set up to describe the foundation of the emotion transition. Theoretically, the model can predict the situations of emotion transition. But it can not support the complicated situations containing certain affective stimulation which happened in common. However, it is necessary to sophisticate the model for both in theory and in practical.

We propose the model with the condition probability tables (CPTs) which consider the external affective stimulations. In the following Fig 2, the model is somewhat like the original one. The arcs represent the transitional probability from one emotion state to another one while the difference is that each circle was replaced by a circle with a inward arrow standing for external affective factors from emotion situation $P(E_k)$.

Since what we are dealing with are in a closed emotion space composed by only seven emotion states, it hypothesizes that state in model is independent from each other. the probability of each emotion situation E_k is $P(E_k)$ and we have $\sum_k P(E_k) = 1\ i = 1 \ldots 7$. The probability of transition from state a_j to state a_i is $P(a_i|a_j, E_k)$.

The CPTs(Conditional Probability Tables) proposed in model indicate the transition probability between two emotions independently. This is actually a Bayesian network model that can be used to infer the emotion in the next period. According to this model, we can learn the probabilities of mental state transition for a person in a particular emotional situation by means of the psychological questionnaire.

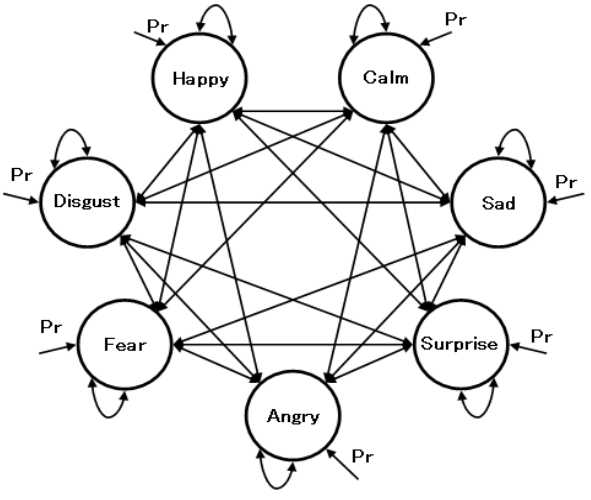


Fig. 2. Advanced Mental State Transition Network Model Considering the External Affective Factors

3 Model Psychological Experiment

3.1 Psychological Experiment Based on the Mental State Transition Network

The conditional probability table is the foundation of the advanced mental state transition network model. In our experiment, CPT is obtained through psychological survey experiment. Over 100 individuals have attended our psychological experiment in the first phase. The participants are recruited primarily from the different high schools and universities in China and Japan. They are from 18 to 30 years old. About 60 percent of them are males and 40 percent are females. All of them were required to fill the questionnaires about emotion state transitions with different clues.

The psychological experiment basically required participants to fill out tables which were designed to describe transitions among seven emotion states following certain clues. The content of the questionnaire is described in the following three parts. First, individual information of the participant, including gender, age, educational level, nationality, occupation, and self-character assess were asked; Second, tables were designed depending on seven discrete mental states and third, an example was presented to show the participants how to fill out the table.

Table 1 is an example of original investigation data we collected. In the table, the header row represents current emotion (mental) state, and the first column represents the emotion (mental) state at next period of time. The digital number in each lattice means the possibility of that situation.

Table 1. Sample of Psychological Questionnaire

	Happy	Calm	Sad	Surprise	Angry	Fear	Disgust
Happy	10						
Calm	8						
Sad	5						
Surprise	3						
Angry	0						
Fear	2						
Disgust	1						

The experiment firstly appealed to the participants to imagine a certain emotional situation under the proposed clues and select what the next emotional state will be with some effect (including internal and external effect). Then we compared each items to calculate the probabilities.

The clues included seven different standard types corresponding to seven prototype emotions. For example, we gave the participants the clue that their wishes suddenly achieved to simulate the happiness situation.

In the questionnaire, the degree of possibility takes an integer value from 0 to 10. The maximum 10 means that the likelihood to transfer from the current state to next state is 100% and the minimum 0 means the possibility of transition is 0%. In table 1, transitional probabilities from the current happy state into happy/ calm/ sad/ surprise /angry/ fear/ disgust states are 10/8/5/3/0/2/1 respectively. In order to allow the participants to fill out the table more easily, the sums of all items in each column are not equal and we must normalize the original data before collecting statistics.

3.2 Model Experiment Result Analysis

Normalization. The original items in the table are designed to be easily filled out and cannot be directly calculated as the probability distribution. Before we collect statistics on the data, we have to normalize the raw data[14].

Table 2. Sample of Transitional Probability in Happy Situation

	Happy	Calm	Sad	Surprise	Angry	Fear	Disgust
Happy	0.443	0.471	0.369	0.426	0.355	0.324	0.383
Calm	0.274	0.259	0.296	0.238	0.276	0.288	0.290
Sad	0.042	0.047	0.099	0.058	0.058	0.058	0.054
Surprise	0.121	0.150	0.147	0.186	0.158	0.145	0.135
Angry	0.029	0.035	0.045	0.048	0.093	0.047	0.049
Fear	0.017	0.021	0.027	0.021	0.038	0.094	0.035
Disgust	0.073	0.018	0.015	0.023	0.022	0.044	0.055

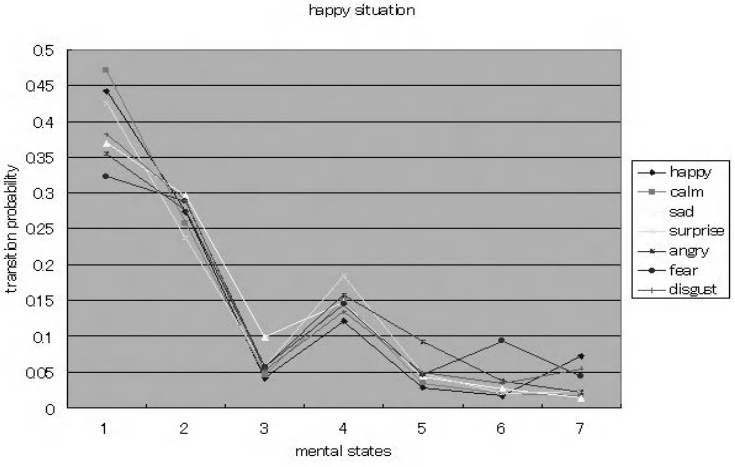


Fig. 3. Conditional Transitional Probability in Happy Situation

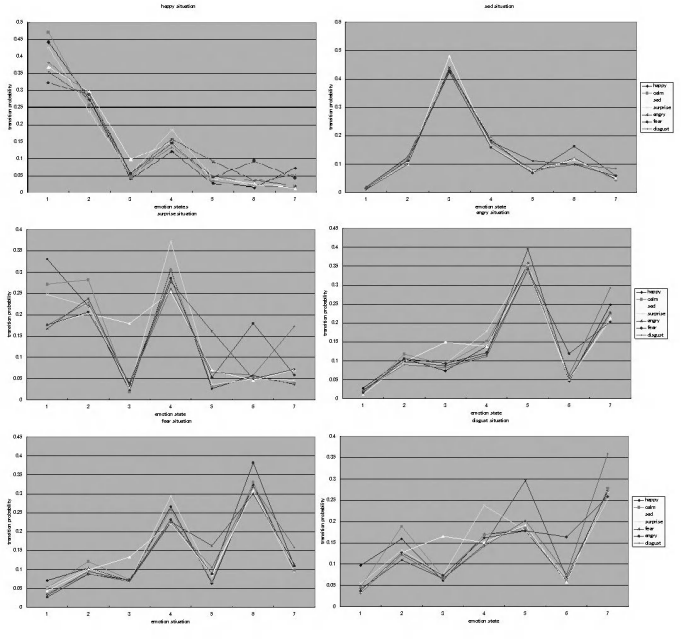


Fig. 4. Conditional Transitional Probabilities in Six Situations

Model Data Analysis. After data normalization, the unbiased estimated means are calculated to obtain the CTPTs(conditional transitional probability tables) of the model. With these CTPTs we can predict the transition procedure of mental states in various situations.

Table 2 includes the unbiased mean of each transition among mental states in the network model. The conditional probability table have been computed in a happiness situation.

Fig 3 indicates the tendency of transitions from different start mental states in the happy situation. From analysis of six emotional situations, it denotes that under external effect:

- The probabilities of all transitions among mental states are all lower than 0.5 in all of six basic emotional situations.
- The mental states are likely to transfer to the states that are similar to the external emotion situations.
- The tendencies of transitions from different states are primarily analogous under a same external environment.

In this way we obtained average of mental state transitional probabilities under conditions of all six prototype emotions that showed in Fig 4.

4 Model Test Experiment

From the previous section a practical transition network model has been built up depending on about 100 pieces of questionnaires. In order to test the validity of this advanced mental state network model, we used another set of 50 random survey results as the test data.

The advanced mental state transition network is used to predict the future emotion state from a previous state with its stationary transitional probability distribution and external condition. Comparison mental states transitions according to the transition network model and test data will certify the validity of the model.

Firstly, in an intuitive viewpoint, we can verify the validity qualitatively. Comparing the top two states transfered from each state between the test data and corresponding probability distribution from the model, the model can be proved to be useful when the states are matching.

Then, we test the model by comparing the transitional probability distribution of all the states. This will finally present a determinate probability that describes the degree of validity of the model.

In the first case, from the model the top two states with largest probabilities are selected to compare with the top two from the test data directly. Table 3 has shown one example result of Comparison in happy situation. The results that the top twp states from the model and test data are matched have indicated that the model is valid qualitatively.

In the second case, the two kinds of transitional probabilities, $P_m\left(a_i|a_j\right)$ and $P_t\left(a_i|a_j\right)$ are considered. $P_m\left(a_i|a_j\right)$ indicates the transitional probability from state a_j to state a_i in model and $P_t\left(a_i|a_j\right)$ is the probability computed from the test data.

$$\sum_i P_m(a_i|a_j) = 1 \tag{1}$$

Table 3. Sample of Qualitatively Test Result of Mental State Transitional Network

	Happy	Calm	Sad	Surprise	Angry	Fear	Disgust
Happy	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Calm	2	2	2	2	2	2	2
Sad							
Surprise							
Angry							
Fear							
Disgust							

$$\sum_i P_t(a_i|a_j) = 1 \tag{2}$$

In our model, there are seven possible states to be transfered into from the start state. In an ideal case, the distribution of the transitional probability of the test data must match the model. We use the difference between distributions of the model and the test data to evaluate the validity. The following equation is used to calculate the related difference between the states:

$$P_j = \sum_i P_m(a_i|a_j)(1 - |P_m(a_i|a_j) - P_t(a_i|a_j)|) \tag{3}$$

The equation describes the difference between the probability distributions, $P_m(a_i|a_j)$ and $P_t(a_i|a_j)$. As the difference increases, the degree of validity P_j decreases. If the distributions of the probability are analogous, the result becomes one. For the whole model, we use the mean value of all states to evaluate the model validity P. The equation is as following:

$$P = \frac{1}{N} \sum_j \sum_i P_m(a_i|a_j)(1 - |P_m(a_i|a_j) - P_t(a_i|a_j)|) \tag{4}$$

N is the total number of the states. P is closer to 1, the model is more valid.

Compared with the 50 random test data, the probabilities of the model validity distributing on the six external emotion situations are indicated in the Tabl 4:

The results mean that the model is close to the actual situation of human mental state transition.

Table 4. Probabilities of the Model Validity

	Happy	Sad	Surprise	Angry	Fear	Disgust
P	0.87	0.85	0.82	0.84	0.85	0.83

5 Conclusion

In paper we proposed an advanced mental state transition network model that can be applied to predict the transition situation of emotion states in the next period under several affective stimulating environments. We implemented these thoughts by the psychological experiments to obtain the conditional transition probability tables in different emotional situations and the validity of the model was tested. The validity test experiment achieved a relatively high precision rate of 0.843 with the set of 50 random test data.

In advanced research, we will expand the range of the psychological experiments and make the model much more accurate and practical through considering more complex emotion states and situations.

Acknowledgment

Our project was partly supported by The Education Ministry of Japan under Grant-in-Aid for Scientific Research B (No. 14380166) and the Outstanding Overseas Chinese Scholars Fund of the Chinese Academy of Sciences (No. 2003-1-1). We also sincerely appreciate all our colleagues participating in this project.

References

1. Ren,F.: Recognize Human Emotion and Creating Machine Emotion. Information. Vol.8 No.1 (2005)ISSN 1343-4500
2. Rosalind W. Picard.: Affective Computing. Preface.pp2. pp22. pp190. The MIT Press Cambridge. Massachusetts London. England (1997)
3. R.Cowie, E. Douglas-Cowie, N. TSApatsoulis, G. Votsis, S. Kollias, W. Fellenz, and J. Taylor.: Emotion Recognition in Human-Computer Interaction. 32-80. IEEE Sig. Proc. Mag., Vol. 18 (1) (2001)
4. P. Ekman, R.W. Levenson and W.V. Friesen.: Automatic Nervous System Activity Distinguishes Among Emotion. Science 221: (1983)1208-1210
5. W.M. Winton, L. Putnam, and R. Krauss. : Facial and Autonomic Manifestations of the Dimensional Structure of Emotion. Journal of Experimental Social Psychology 20: (1980)195-216
6. N.H. Frijda.: The Emotion. The studies in Emotion and Social Interaction. Cambridge University Press Cambridge (1986)
7. Plutchik, R.: Emotions: A Psychoevolutionary Synthesis. New York: Harper Row. (1980)
8. Russell,J.A.: A Circumplex Model of Affect. Journal of Personality and Social Psychology 39. (1980)1161-1178
9. P. R. Kleinginna., Jr. and A. M. Kleinginna.: A Categorized list of Emotion Definitions. with Suggestions for a Consensual Definition. Motivation and Emotion. 5(4)(1981)345-379
10. Xiang,H., Jiang,P.,Ren,F.,Kuroiwa,S.: An Experimentation on Creating a Mental State Transition Network IEEE. Proc.ICIA. Hong Kong. China. (2005)432-436
11. P,Ekman.: Universals and Cultural differences in Facial Expressions of Emotion. In J.Cole (Ed.) Nebraska Symposium on Motivation. Vol.19. Lincoln: University of Nebraska Press. (1972)207-283

12. Goldstein, MD., Strube MJ.: Independence Revisted The Relation between Positive and Negative Affect in A Naturalistic Setting. Pers. Soc. Psychol. Bull. 20 (1994) 57-64
13. Oatley, K., Jenkins, J.M.: Understanding Emotions. Blackwell (1996)
14. Yamata Takeshi, Murai Junnichiro : Yokuwakaru Sinritookei (Understanding psychological statistic) ISBN 4-623-03999-4 (2004) 32-80

Development of a Microdisplay Based on the Field Emission Display Technology

Takahiro Fusayasu[1], Yoshito Tanaka[1], Kazuhiko Kasano[2], Hisashi Fukuda[3], Peisong Song[1], and Bongi Kim[1]

[1] Nagasaki Institute of Applied Science
[2] Display Tech 21, Inc.
[3] Muroran Institute of Technology

Abstract. We have been developing a microdisplay based on the field emission display (FED) technology, which is advantageous in power consumption, image quality and long term stability. We have adopted LSI-driven anode pixels, which enables active-matrix addressing and, therefore, highly precise and high-quality microdisplay. The structure was optimized according to the simulation study of electric field and electron trajectories. The driver LSI has been designed, evaluated by simulation, and the wafers have been produced. Anti-crosstalk grid should be constructed on the LSI by photolithography and the relevant study has been performed.

1 Introduction

Microdisplays are defined as microminiaturized displays typically with screen size less than 1.5" diagonal. They are used in wearable displays, in the traditional viewfinders of digital cameras and in mobile communication instruments such as cellular phones. While ubiquitous instruments are expanding in our life, demand for the microdisplay is rapidly increasing.

In Table 1, technologies to realize microdisplays are shown. Although the cathode ray tube (CRT) display provides high image quality, it is not regarded as most suitable for microdisplay due to its large power consumption and size. The liquid crystal display (LCD), which is the most widely used flat panel display (FPD), is also not most advantageous in power consumption and volume because it needs a backlight. The Organic LED (OLED), recently emerging as a thriving technology for flexible FPDs, is still on its way to overcoming the short lifetime. It can be suitable for cellular phones since many of customers replace their cellular phones to new products frequently. Microdisplays for the wearable displays, however, would be required to have enough endurance especially when they are subjected to outdoor uses. The field emissions display (FED) is expected to involve high image quality of CRT, small volume, low power consumption and long term stability. Therefore, we have chosen the FED technology to develop a microdisplay.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 1036–1044, 2005.
© IFIP International Federation for Information Processing 2005

Table 1. Microdisplay technologies compared to our method (FED). Comparison is made for XGA (1024 × 768 pixels) displays. The latest models in 2003 are referred except for the FED.

Technology	Voltage	Power Consumption	Light Emission	Size ($W \times L \times H$ [mm])
CRT	high	1500mW	self-emissive	$30 \times 75 \times 19$
LCD	low	900mW	backlight	$40 \times 38 \times 19$
OLED	low	300mW	self-emissive	$26 \times 20 \times 10$
FED	low	50mW	self-emissive	$26 \times 23 \times 5$

2 Structure of the Field Emission Microdisplay

Fig. 1(a) shows the most commonly used structure of the FED. Electrons are emitted from field emitters that are arrayed in pixels on a cathode plate. Carbon nanotubes (CNTs) [1] are often used as the field emitters because their high aspect ratios, good chemical stability and high mechanical strength are thought to be most suitable for stable field emitters with low threshold voltage in FEDs [2]. The emitted electrons collide with the phosphor painted on the anode plate and cause electron-beam induced emission of light. The electron current of each pixel is controlled by voltages of the gate and cathode electrodes which cross at that pixel.

The structure design of our microdisplay is illustrated in Fig. 1(b). Here, CNTs are put on one cathode plate and applied same voltage. The gate voltage is also fixed at one value. The CNTs, the gate and the mesh grid compose a triode structure, which introduces field emission from the CNTs. The electrons go through the mesh grid and reach the [ZnO:Zn] phosphor painted on the anode pixels, which consist of surface metal of a CMOS LSI. The brightness of each pixel is determined by the anode voltage, which is controlled by the pixel driver LSI. Additional grid electrodes are necessary between the pixels such that crosstalks should be suppressed. The emitted light transmits through the LSI, because the LSI is grinded to the thickness of about 30μm, and filtered by the red, green and blue color filters.

The most characteristic point of this structure is that the brightness is controlled by the anode pixels. This method has following advantages:

1. Active-matrix addressing is possible because the driving LSI is free from thermal damage caused by CNT synthesis. The cathode-driven FED, in which passive-matrix addressing is usually used, often suffers from parasitic-capacitor induced crosstalk.
2. Driving voltage is small and therefore the production cost can be decreased. In the case of the cathode-driven FED, large voltages applied to the gate and the cathode must be driven.
3. The cathode just provides uniform distribution of electrons and needs not to be focused. On the other hand, electrons in the cathode-driven FED must be focused onto the facing anode pixel, and the focus ability limits down-sizing.

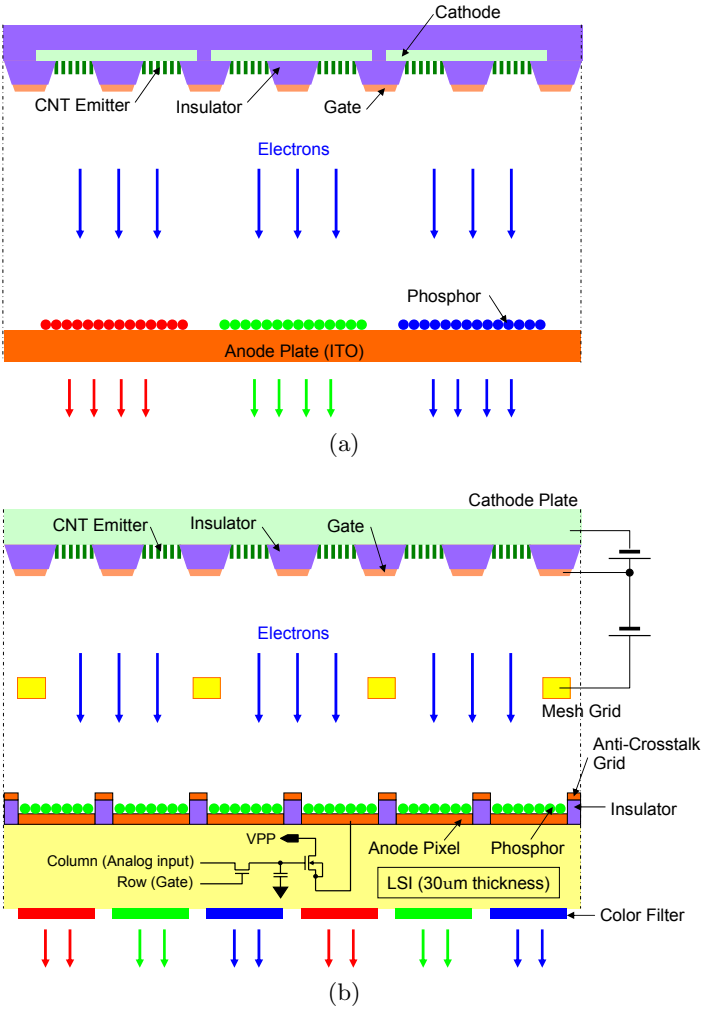


Fig. 1. (a) The structure of the most commonly used FED design, where the brightness of each pixel is controlled by the anode and gate voltages. (b) The structure of our field emission microdisplay, where the pixel brightness is controlled by the anode pixels.

4. Since the cathode does not have pixel structure, precise alignment between the cathode and the anode is unnecessary and the production process can be simplified. In addition, the anode and the cathode can be developed individually.

3 Simulation of Electric Field and Electron Trajectories

In order to design the structure shown in Fig. 1(b), we have investigated electric field and electron trajectories for various candidate structures using the two-

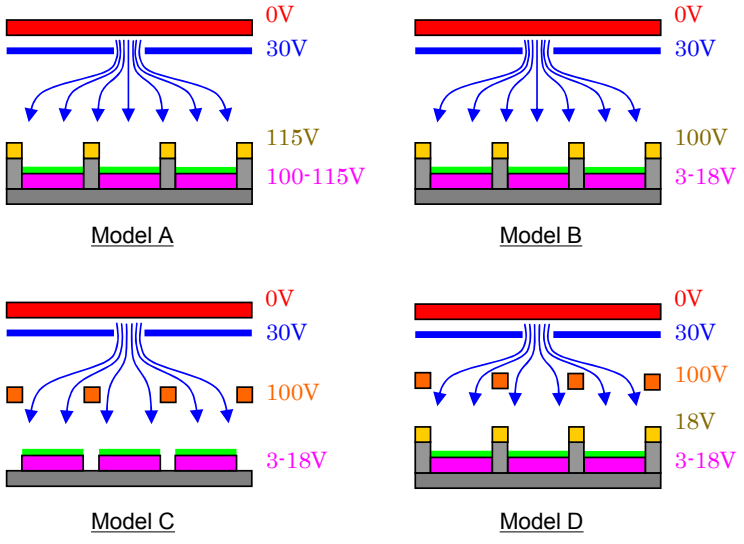


Fig. 2. Models used in the simulation of electric field and electron trajectories

dimensional finite element method (FEM) software package, "TriComp". The examined models are illustrated in Fig. 2. Distance between cathode and anode plates is supposed to be 500μm for the models A and B, and 1000μm for the models C and D. For the models C and D, a floating mesh grid is inserted in the midst of between the cathode and anode plates. The pixel size is supposed to be 8μm width and the grid width to be 2μm. Since the LSI is produced by a 16V process, the maximum amplitude of the anode voltage is about 15V, where the CMOS threshold voltage of about 1V is subtracted.

In the model A, the anode LSI chip is biased 100V higher than the cathode so that the cathode CNTs can be applied enough field. In this case, energy of arriving electron only differs by 15%, i.e. between 100eV and 115eV. In addition, the simulation has shown that the change of the number of electrons is also very small. Therefore, we cannot expect enough contrast for this model.

In the model B, in order to circumvent the above problem, the minimum voltage of the anode pixel is set 3V higher than the cathode voltage so that the minimum-energy electrons reach the phosphor with its emission threshold energy of about 3eV. On the other hand, the on-LSI grid is set to 100V, which is necessary for the electron emission by CNTs. Here, electrons are accelerated towards the on-LSI grid and then decelerated towards the anode pixels. In this model, many part of electrons were found to be absorbed in the on-LSI grid or collide with the sustaining insulator because of strong field generated around there and may cause a charge-up problem.

The model C has been suggested so that the charge-up is suppressed by separating the grid from the LSI. However, the model C suffers from crosstalk between pixels because of absence of the on-LSI anti-crosstalk grid.

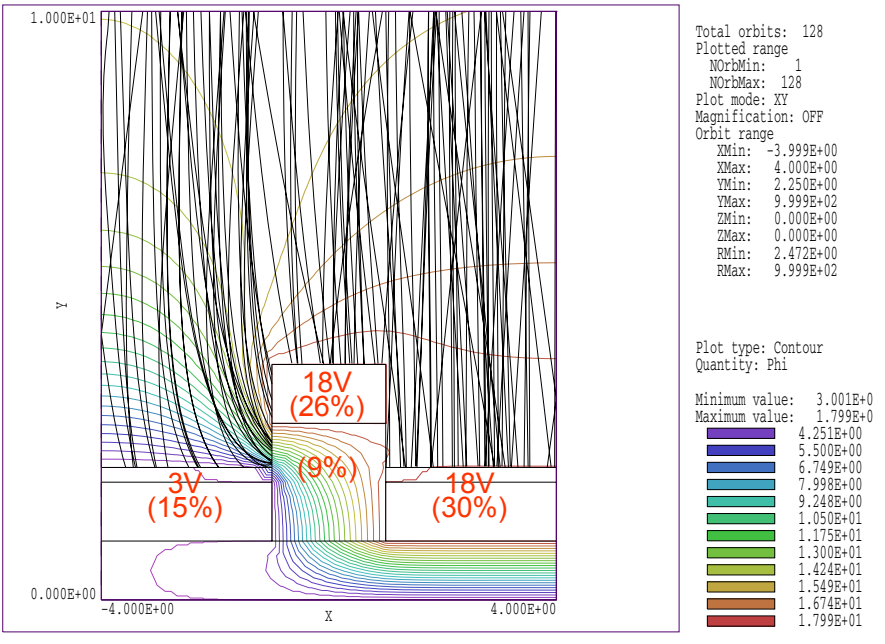


Fig. 3. Field and electron track simulation results based on the model D when the left pixel is set to 3V and the right pixel to 18V. The grid is set to the same voltage as the maximum allowed voltage for the pixels. Only around the LSI surface is shown. Also shown are the ratio of the number of electrons against the total generated electrons.

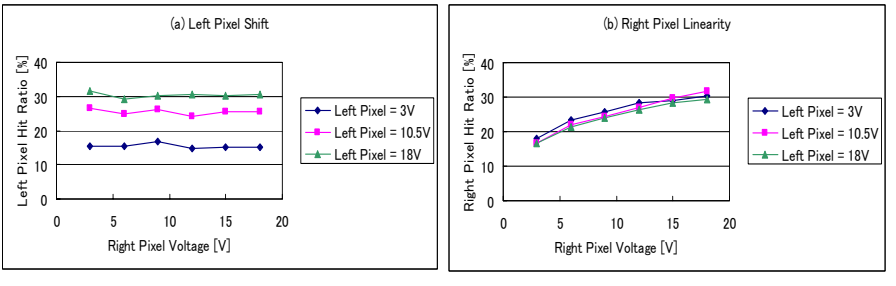


Fig. 4. Estimation of crosstalk between neighboring pixels based on the model D

Finally, we propose the model D, the double grid structure. The simulation result based on the model is shown in Fig. 3. As can be seen, the number of electrons arriving the insulator is small enough (about 9% of emitted electrons). Fig. 4 shows the change in the number of electrons arriving the anode pixels according to the anode voltages. Here, the left pixel voltage is fixed and the right pixel voltage is swept. Changes observed on the left pixel, as shown in Fig. 4(a), are flat within the size of statistical error and indicate that the crosstalk between

the neighboring pixels is small enough. It is found that the on-LSI grid should be set taller than the phosphor surface in order that the grid works well. The smooth increase for the right pixel, shown in Fig. 4(b), would help deep contrast and good linearity of brightness against the applied voltage.

4 Design of the Driver LSI Chip

The LSI chip to drive the anode pixels is designed based on the high-voltage 0.6μm CMOS technology. The device provides a matrix output of open-source high-voltage N-channel MOSFETs and is equipped with 352×240 pixels on its face. The analog and digital power supplies are both 16V and the clock speed is 7.16MHz. Each pixel has a size of $30 \times 33\mu$m and consists of three sub-pixels, as can be seen from the pixel layout in Fig. 5(a). Larger area is devoted to the red pixels than others because the red light would be weaker than green and blue lights after passing the color filters. The sub-pixel circuit component is illustrated in Fig. 5(b). The analog voltage is supplied from the COL* input at the certain clock period scheduled, and held in the capacitor by the latch signals ROW* and ROWX*. During the frame period of 32ms, the held voltage is applied to the anode pad via a source follower. The droop is designed to be less than 20% of the held voltage during the frame period.

The Fig. 6 shows the block diagram of the driver LSI. The analog data for red, green and blue sub-pixels are fed in parallel into the column registers. The column resisters COL1 to COL352 are opened in turn according to the CCLK with the period of 140ns. Then, the signals are gated by the row register outputs ROW(X)1 to ROW(X)240, which are controlled by the RCLK with the period of 63.5ns.

The design has been submitted to the LSI processing company and prototype wafers have been delivered. We are now preparing for evaluation of electrical properties of the chip.

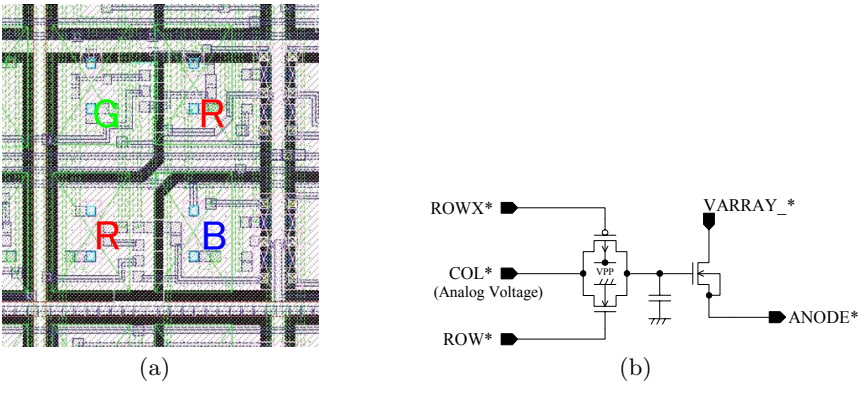


Fig. 5. (a) The pixel layout design and (b) the sub-pixel circuit component

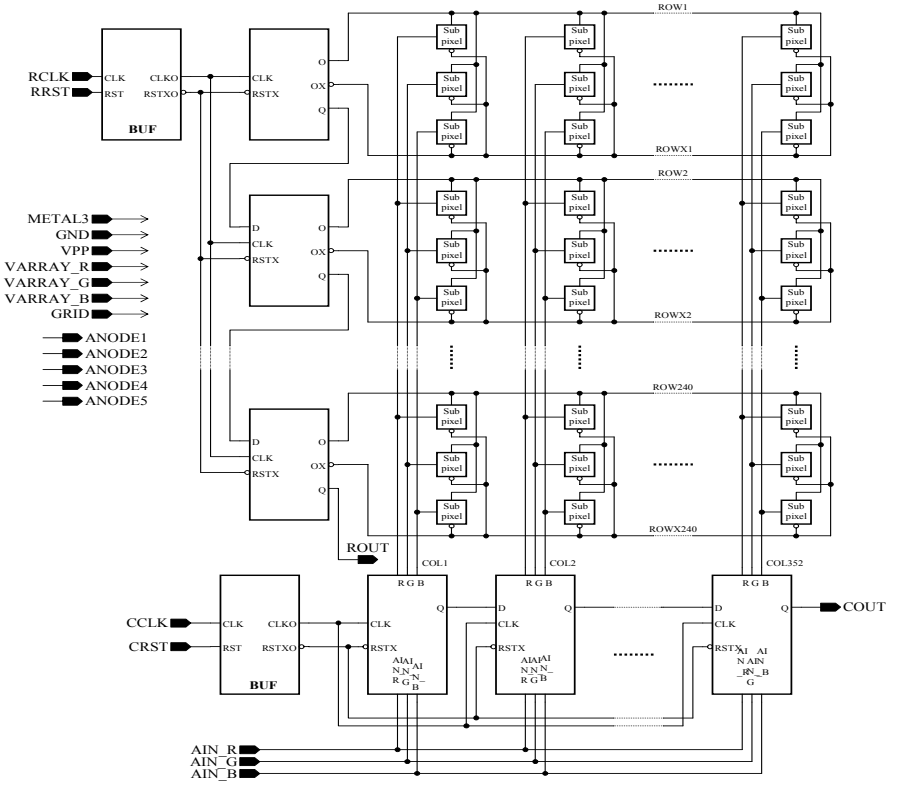


Fig. 6. The block diagram of the driver LSI

5 Post-processes on the Anode LSI

As described in Section 3, anti-crosstalk grids need to be placed between the pixels together with insulator pillars underneath. The process to construct the grid structure is shown in Fig. 7. On the LSI chip (1), a SiO_2 layer is created by the thermal CVD method (2), and Cr and Au films are coated on it by spattering (3)-(4). Here, the Cr layer is interleaved as glue so that the Au would not peel off when resist is removed afterwards. Next, photo-resist is coated on the Au layer and patterned by photolithography (5)-(6). Then, the Au layer is patterned by wet etching (7) followed by the Cr etching (8) and the resist is removed (9). Finally, the silicon-oxide layer is patterned by dry etching with the metal layers used as a mask.

For the purpose of test, we have first tried the above process on a small silicon piece substrate. The scanning electron microscopy (SEM) image of the piece observed after the process (9) is shown in the upper left picture in Fig. 8. Note that, for the purpose of investigating the minimum possible grid width, the pattern used here is different from the one for the target LSI.

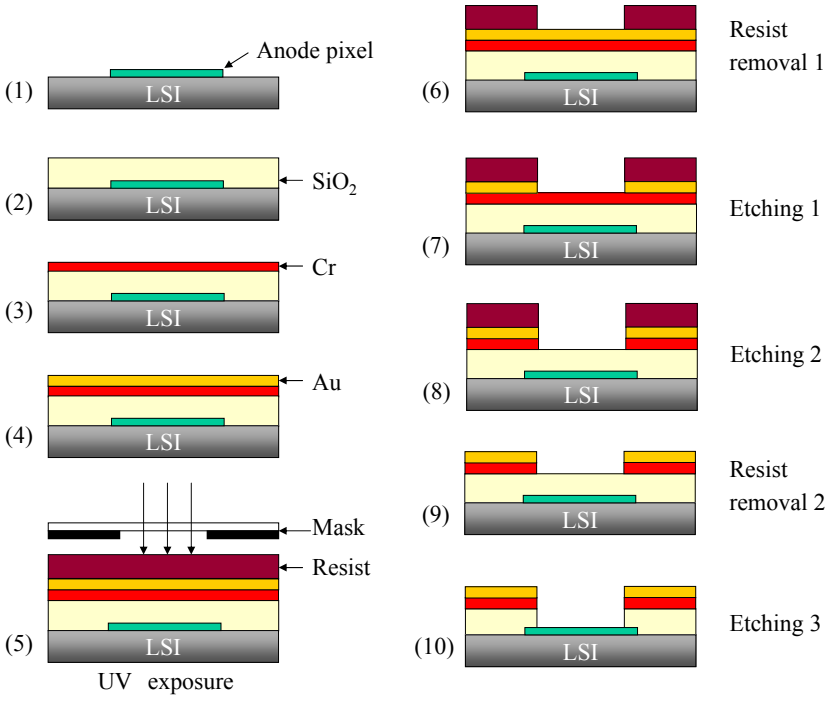


Fig. 7. The anode process to create the on-LSI anti-crosstalk grid

In order to confirm the right patterning of each layer, energy-dispersive spectroscopy (EDS) mapping has been performed. The upper right picture in Fig. 8 shows the EDS mapping image of oxygen, which actually indicates the SiO_2 image. Here, since the SiO_2 layer is shaded by the lattice-shaped upper layers, only the X-ray coming through the windows are observed. The lower-left picture illustrates the image of the Cr layer, which seems to be etched as designed, though the contrast is worse due to the X-ray absorption by the upper Au layer. The lower-right picture shows the image of the Au layer, which is also found to be patterned as designed.

6 Summary and Future Development Plans

We have been developing a microdisplay based on the FED technology with the anode-driven method. The structure was optimized by the simulation study of electric field and electron tracks. The double grid structure was found to be the best. The driver LSI was designed and the prototype wafers were produced. Their characteristics are to be evaluated soon. The anode post-process for construction of the on-LSI grid was proposed and evaluated using a small Si piece, which is imitation of the LSI chip. We will soon test the process using the real LSI chip.

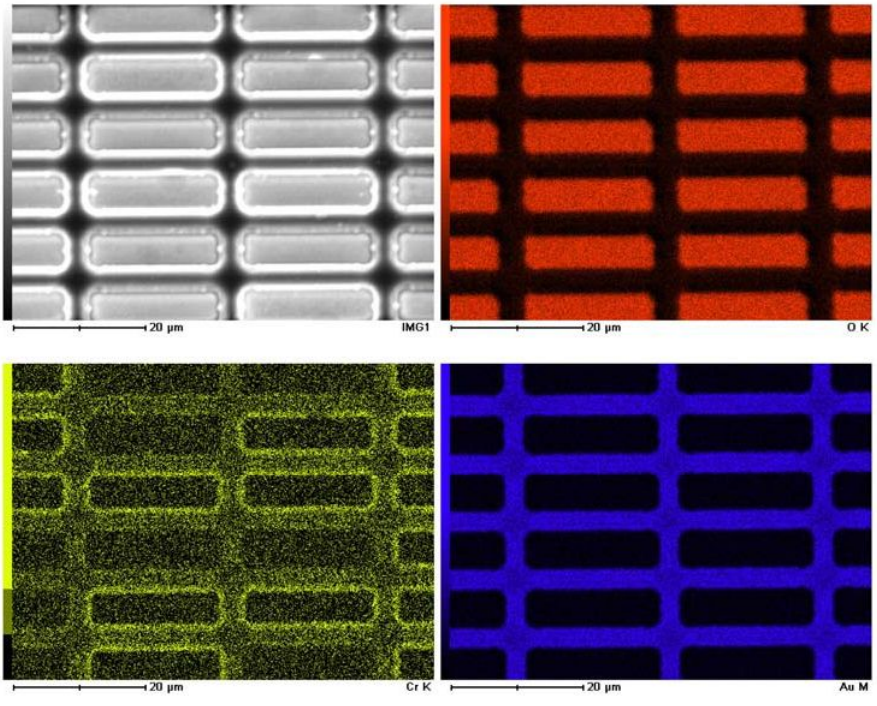


Fig. 8. The SEM image of the test piece after the resist removal (upper left), EDS mapping images of oxygen (upper right), chromium (lower left) and gold (lower right)

Patterning of the phosphor should also be developed. In addition to the studies introduced in this paper, we have been investigating about the CNT cathode. Accordingly, construction of the complete microdisplay system will be realized in the near future.

Acknowledgements

We thank Mr. Tomoaki Sugawara at Hokkaido Industrial Technology Center for his vital support on the semiconductor processes. The driver LSI was produced by Magna Chips. The project is supported by Japan Science and Technology Agency (JST).

References

1. S. Iijima, Nature **354** (1991) 56.
2. W. A. de Heer, A. Chatelain and D. Ugarte, Science **270** (1995) 1179.

Information Flow Security for Interactive Systems*

Ying Jin[1], Lei Liu[1,**], and Xiao-juan Zheng[2]

[1] Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education of P.R. China, Computer Science & Technology College, Jilin University, Changchun, 130012, P.R.China
[2] Software College, Northeast Normal University, Changchun, 130117, P.R. China
liulei@mail.jlu.edu.cn

Abstract. The use of the Internet raises serious behavioural issues regarding, for example, security and the interaction among agents that may travel across links. Model-building such interactive systems is one of the biggest current challenges in computer science. A general model, action calculi, has been introduced by Robin Milner to unify the various emerging disciplines of interactive behaviour. In this paper action calculi is used as an abstraction of interactive systems and information flow security properties of such systems are studied. At first an information flow analysis for static action calculi is presented to predict how data will flow both along and inside actions and its correctness is proved; Next basing on the result of the analysis information security properties of both static and dynamic action calculi are discussed; Finally a general relationship are established between the static notation of information flow security and the dynamic one.

1 Introduction

The use of the Internet raises serious behavioural issues regarding, for example, security and the interaction among agents that may travel across links. Model-building such interactive systems is one of the biggest current challenges in computer science. Action calculi have been introduced by Milner[6, 8] as a framework for representing models of interactive computation. It is indicated in [6, 8, 10, 11] that action calculi show the advantage in uniting different models of interactive systems in a common setting[6, 8, 10, 11], and such unification is necessary for studying general properties such as security properties of these systems[5].

Information flow security is concerned with controlling the flow of information within a sytem. Program analysis such as information flow analysis aims at verifying properties of a program that hold in all execution, which recently has been used for validating security and safety issues for concurrent and interactive systems[16]. In this paper, we use action calculi as an abstract of interactive systems, and study its in-

* This paper is sponsored by Jilin Province Science and Technology Development Plan Project (No. 20050527).
** Corresponding author.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 1045 – 1054, 2005.
© IFIP International Federation for Information Processing 2005

formation security properties basing on information flow analysis. The molecular forms of action calculi give normal form for the algebraic terms and suggest a modular style of programming and system description. Therefore, we propose a formal information flow analysis for the molecular forms of static action calculi, which statically shows how data will flow both along and inside molecules. In order to facilitate the formulation of the analysis, we make trivial extension to the syntax of action calculi in that giving every action and molecule a unique name. Following the idea of [1, 2], we state simple security property for static action calculi, that is: the secrecy of data is preserved if an action a never read an untrustworthy name to it or an action a never write to other actions the untrustworthy name. Finally we introduce security property for dynamic action calculi, and establish the general relationship between the static notation and dynamic one.

The rest of the article is organized as follows. Section 2 briefly reviews the basic concepts and definition of action calculi. Section 3 presents formalization of information flow analysis of static action calculi and proof of the correctness of the analysis. Security properties basing on the analysis above and a general relationship between the static notation and dynamic one are established. Finally related work and conclusion are addressed in section 5.

2 Preliminaries

In this section we briefly review the definition of action calculi following [6,7, 8]. A trivial extension is made by adding unique identifier to every action and every molecule.

In this paper we focus on action calculi in molecular forms. An action calculus is determined by a signature K = (P, K) together with a set of control rules. K consists of a set P of basic types, called primes and denoted by p, q, …, and a set K of constants, called controls. Each control in K has an associated arity ((m1, n1), …, (mr, nr)) → (m, n), where the m's and n's are finite sequences of primes, called tensor arities; we write ε for empty sequence, ⊗ for concatenation using infix notation, and write M for the set of tensor arities.

Definition 1. (Molecules and molecular forms) Let $\mathbb{K}$ be a signature. The molecular forms over $\mathbb{K}$ are syntactic objects; they consist of the actions *a* defined as follows, in terms of molecules μ:

$a ::= A[(\bar{x})\ \mu_1 \ldots \mu_r < \bar{u} >]$ $\qquad (\bar{x} : m.;\ \bar{u} : n\ ;\ a : m \rightarrow n)$

$\mu ::= M[< \bar{v} > K\ \bar{b}\ (\ \bar{y}\ )]$ $\qquad (\bar{v} : k\ ;\ \bar{y} : l\ ;\ K\bar{b} : k \rightarrow l)$

where A and M are identifiers of action and molecule respectively, the sequence μ_1, …, μ_r is called the body of *a*, K is a control, $(\bar{x})$ and $(\bar{y})$ are called *imported names* (which must be distinct) denoted as *imp*(*a*/μ), $< \bar{u} >$ and $< \bar{v} >$ are called *exported names* denoted as *exp*(*a*/μ).

Definition 2. (subaction) The subactions of an action *a* comprise *a* itself and the subactions of each action in a molecule of *a*.

For simplicity, we will use identifier to represent corresponding action or molecule.

In action a the imported names $\bar{x}$ are binding, and the names $\bar{y}$ of the molecule are also binding. The scope of each binding extends to its right, to the end of the smallest subaction containing it. Molecules are binding operators. In the above molecule μ, the names $<\bar{v}>$ occur free; they are the means by which it is bound into an action. In the above action a, any name-vector in round brackets – either at the head of a or at the right end of a molecule in μ - is binding, and its scope extends rightwards to the end of a.

Definition 3. (Operations over molecular forms) The operations $\mathbf{id}_k$, ·, ⊗, ab_x, <x> and ω are defined over molecular forms. The detailed definitions are in [6, 7, 8].

Definition 4. (Action calculi: static) A static action calculi comprises a signature $\mathbb{K}$, together with a set of actions in the molecular forms over $\mathbb{K}$, and a set of operations defined as above. We call this static action calculus over $\mathbb{K}$, and denote it by $AC^s(\mathbb{K})$.

Definition 5. (Control operations) Each control *K*, is defined as a control operation upon molecular forms as follows:

$$K(a) \stackrel{\text{def}}{=} A_n[(\bar{x})<\bar{x}>Ka(\bar{y})<\bar{y}>] \qquad (\bar{x}, \bar{y} \text{ not free in } a)$$

Definition 6. (Control rule) A control rule over a signature $\mathbb{K}$ takes the form

$$t[\bar{a}] \searrow t'[\bar{a}]$$

where t and t' are terms built from metavariables $\bar{a}$ using controls together with the operations in definition 4.

Definition 7. (Action calculi: dynamics) A dynamic action calculus comprises a signature $\mathbb{K}$ and a set of *R* of control rules over, together with the static action calculi $AC^s(\mathbb{K})$ equipped with the smallest reaction relation $\searrow$ which satisfies the rules *R* (for all replacements of metavariables *a* by actions). We call this the dynamic action calculus over $\mathbb{K}$ and *R*, and denote it by $AC(\mathbb{K}, R)$.

3 Information Flow Analysis

3.1 Notations and Definitions

A *path* is an ordered list of identifiers for actions or molecules. An *occurrence path* for an action *a* or a molecule *m* or a name *x* in an action *a'* or a molecule *m*' is a path consisting of all the identifiers for those actions or molecules along which the action/molecule/name could be reached, which is denoted as *Path*(*a*/*m*/*x*, *a'*/*m'*).

We also define *path environment*, *abstract binding environment* and *abstract bound environment* as following:

- σ is the path environment that associates imported names with the occurrence path of that name in the corresponding molecular forms; Current path environment contains those names that are scope effective in current action or molecule together with corresponding paths; For a name x and its path p, σ[x→p] represents updating σ by modifying the path of x as p;
- ρ is the abstract binding environment that associate a given export name x and its occurrence path p with the occurrence path of the binding name. This means that ρ(x,p) returns the occurrence path of the imported name x which is binding the exported name x occurring on the path p; For a name x and its path p,

$\rho[(x,p)\rightarrow p']$ means adding ρ with a new binding (imported name x of path p' binds exported name of path p);

- κ is the abstract bound environment that associates a given import name x and its occurrence path p with the set of occurrence paths of the exported names bound by corresponding x. More precisely, for a given export name x and its occurrence path p, $\kappa(x,p)$ returns the paths of all the name bound by x; For a name x and its path p , κ $[(x,p)\rightarrow P']$ means adding κ with a new bound (imported name x of path p binds a set of exported names of every path in P').

Path environment, binding environment and bound environment together are called abstract environment.

We define an operation $\vee$ for unifying abstract binding and bound environment:

$$\rho_1 \vee \rho_2 (x, p) = \begin{cases} \rho_1(x, p), & \text{if } \rho_1(x, p) \neq () \\ \rho_2(x, p), & \text{if } \rho_2(x, p) \neq () \\ () , & \text{otherwise} \end{cases}$$

$$\kappa_1 \vee \kappa_2 (x, p) = \kappa_1(x, p) \cup \kappa_2 (x, p)$$

3.2 Strategy of Information Flow Analysis

Information flow analysis for action calculi is presented and its correctness is proved.

The aim of information flow analysis for static action calculi is to determine which imported names in a certain action or molecule will bind which exported names. This result will be used to check the information flow along internal actions so as to prove whether such information flow is secure under security policy. The result of static analysis of the syntactic molecular forms include:

a) the binding imported name for any exported name;
b) all bounded exported names for any binding imported name x;

A function performing information flow analysis is defined. For given molecular form of an action or a molecule with its path, the function accepts current abstract environment and computes new abstract environment.

The definition of the function is as follows:

$[m]p\sigma\rho\kappa = (\sigma', \rho', \kappa')$, where m is the molecular form of an action or a molecule, p is the path for m, $(\sigma', \rho', \kappa')$ is the current abstract environment, and $(\sigma', \rho', \kappa')$ is the new abstract environment.

The function is defined on an action in the form of $id[(\bar{x})<\bar{y}>]$ and $id[(\bar{x})\ [\mu 1 \ldots \mu r]<\bar{y}>]$, a molecule in the form of $id[<\bar{u}> K (\bar{v})]$and $id[<\bar{u}> Ka (\bar{v})]$. The analyzing process is based on nesting scope rules for static action calculi. The detailed function definition and explanation are as follows.

$[\![id[(\bar{x})<\bar{y}>]]\!]\ p\sigma\rho\kappa =$

let $\sigma' = \sigma[x_i \rightarrow p]$ (i = 1, …, m)

in *let* $\rho_0 = \Phi, \kappa_0 = \Phi$

in *let* for i = 1 to n do

$$\{ \text{ if } y_i \in \overline{x} \text{ then } \{ \rho_i = \rho_{i-1}[(y_i,p)\rightarrow p];\ \kappa_i=\kappa_{i-1}[(y_i,p)\rightarrow\{p\}];\}$$
$$\text{else if } \sigma'(y_i)\neq\{\} \quad \text{then } \{\rho_i= \rho_{i-1}[(y_i,p)\rightarrow\sigma'(y_i)];$$
$$\kappa_i=\kappa_{i-1}[(y_i,p)\rightarrow\kappa_{i-i},p)\cup\{p\}];\}$$
$$\text{else } \{ \rho_i = \rho_{i-1}[(y_i,p)\rightarrow()];\ \kappa_i = \kappa_{i-1}\}$$
$$\underline{in}\ (\sigma', \rho \vee \rho_n, \kappa \vee \kappa_n)$$

For simple actions with empty body, first we would update path environment with every imported name and its path as current path p, then for every exported name, we will check if there is corresponding name in binding environment, if so a new binding should be added to binding environment and put the path of this exported name into the set of bound path of the binding name, otherwise the exported name is a free name, so we would add an empty path as the path of its binding and make bound environment unchanged.

$$[\![\mathrm{id}[(\overline{x})\ [\mu_1 \ldots\mu_r]<\overline{y}>]]\!]p\sigma\rho\kappa =$$
$$\underline{let}\ \sigma_0= \sigma[x_i\rightarrow p]\ (i = 1, \ldots, m)$$
$$\underline{in}\ \underline{let}\ \rho_0 = \Phi,\ \kappa_0 = \Phi$$
$$\underline{in}\ \underline{let}\ \text{for } i = 1 \text{ to } r \text{ do}$$
$$(p_i, \sigma_i, \rho_i, \kappa_i) = [\![\mu_i]\!](\mathrm{cons}(p,\mathrm{id}(\mu_i)), \sigma_{i-1}, \rho_{i-1}, \kappa_{i-1})$$
$$\underline{in}\ \underline{let}\ \rho_0 = \rho_r,\ \kappa_0 = \kappa_r$$
$$\text{for } j = 1 \text{ to } n \text{ do}$$
$$\text{if } \sigma_r(y_j) \neq\{\} \text{ then } \{\rho_j= \rho_{j-1}[(y_i,p)\rightarrow\sigma'(y_j)];$$
$$\kappa_j=\kappa_{j-1}[(y_i,p)\rightarrow\kappa_{j-1}(y_j,p)\cup\{p\}];\}$$
$$\text{else } \{ \rho_j = \rho_{j-1}[(y_j,p)\rightarrow()];\ \kappa_j = \kappa_{j-1};\}$$
$$\underline{in}\ (\sigma_r, \rho \vee \rho_n, \kappa \vee \kappa_n)$$

For actions with its body as a nonempty sequence of molecules, first we would also update path environment with every imported name and its path as current path p; then in order to apply for the binding mechanism of molecules, for every molecule we will sequentially compute their result abstract environment with former result abstract environment as input. Finally we will compute on the exported names of the action. Here we use cons(p, id(μi)) represent the concatenating current path with current molecule identifier to get the path for this molecule.

Analyzing molecule also includes two computation forms:

$$[\![\ \mathrm{id}[<\overline{u} > K\ (\overline{v})]]\!]p\sigma\rho\kappa=$$
$$\underline{let}\ \rho_0 = \Phi,\ \kappa_0 = \Phi$$
$$\text{for } j = 1 \text{ to } n \text{ do}$$
$$\text{if } \sigma(u_j) \neq\{\} \text{ then } \{\rho_j= \rho_{j-1}[(u_i,p)\rightarrow\sigma'(u_j)];\ \kappa_j=\kappa_{j-1}[(u_i,p)\rightarrow\{p\}];\}$$
$$\text{else } \{ \rho_j = \rho_{j-1}[(u_j,p)\rightarrow()];\ \kappa_j = \kappa_{j-1};\}\ \ \sigma'= \sigma[v_i\rightarrow p]\ (i = 1, \ldots, m)$$
$$\underline{in}\ (\sigma', \rho \vee \rho_n, \kappa \vee \kappa_n)$$

For molecules with constant control, first we would also update binding environment and bound environment by computing on their exported names with respect to input abstract environment, then update path environment with every imported name and its path as current path p.

$$
\begin{aligned}
&[\![\mathrm{id}[<\bar{u}> \mathrm{Ka}\,(\bar{v})]]\!]\mathrm{p}\sigma\rho\kappa = \\
&\qquad \underline{let}\ \ \rho_0 = \Phi,\ \kappa_0 = \Phi \\
&\qquad \text{for } j = 1 \text{ to } k \text{ do} \\
&\qquad\qquad \text{if } \sigma(y_j) \neq \{\} \text{ then } \{\rho_j = \rho_{j-1}[(u_i,p)\rightarrow\sigma'(y_j)];\ \kappa_j = \kappa_{j-1}[(u_i,p)\rightarrow\{p\}];\} \\
&\qquad\qquad\qquad \text{else } \{\ \rho_j = \rho_{j-1}[(u_j,p)\rightarrow()];\ \kappa_j = \kappa_{j-1};\} \\
&\qquad \underline{in}\ \ \underline{let}\ \ (\sigma_1, \rho_1, \kappa_1) = [\![a_1]\!]\,(\mathrm{cons}(p,\mathrm{id}(a_1)), \sigma', \Phi, \Phi) \\
&\qquad\qquad \cdots\cdots \\
&\qquad\qquad (\sigma_n, \rho_n, \kappa_n) = [\![a_n]\!]\,(\mathrm{cons}(p, \mathrm{id}(a_n)), \sigma', \Phi, \Phi) \\
&\qquad\qquad \underline{in}\ \ \underline{let}\ \ \sigma' = \sigma[v_i\rightarrow p]\ (i = 1, \ldots, l) \\
&\qquad\qquad \underline{in}\ \ (\sigma', \rho_k \vee \rho_1 \vee \ldots \vee \rho_n,\ \kappa_k \vee \kappa_1 \vee \ldots \vee \kappa_n)
\end{aligned}
$$

For molecules with nonzero rank control, first we would also update binding environment and bound environment by computing on their exported names with respect to input abstract environment. Then since the scopes of the action parameters of a molecule are independent, we will update their result abstraction environment in parallel with the same input abstract environment. Next path environment will be obtained by updating original input path environment with every imported name and its path as current path p because the imported names in action parameters will not be effective outside the molecule. Finally the resulting binding environment and bound environment are the combination of all the binding environment and bound environment obtained from the computations of those action parameters.

3.3 Correctness

The correctness of the function is demonstrated by showing that the function can compute correct abstract environment for any molecular forms.

Here with four prepositions we get one lemma which will show that with the above function we can get correct abstract environment for any molecular forms.

Proposition 1. Assume that $\mathrm{id}[(\bar{x})<\bar{y}>]$ is a subaction of an action *a*, p is its path in *a*, σ is the right set of names and their paths that is effective for id, if $[\![\mathrm{id}[(\bar{x})<\bar{y}>]]\!]\mathrm{p}\sigma\rho\kappa = (\sigma', \rho', \kappa')$ then we have :

(1) yi is bound by an imported name of path p' in a iff $\rho'(p, yi) = p'$;
(2) xi is binding an exported name of path p' in a iff $p' \in \kappa'(p, xi)$.

Proof: From the definition of $[\![\mathrm{id}[(\bar{x})<\bar{y}>]]\!]$we can see that:

(1) σ' is just obtained by updating σ with current new effective imported names x; since σ is right, therefore σ' is right too;
(2) with new computed σ' we can compute (ρn, κn) by adding exported names y; since σ' is right, therefore from the computation process of (ρn, κn) we can say

that (ρn, κn) is correct for id, therefore adding (ρn, κn) to original (ρ, κ) will produce a correct one (ρ' , κ')so far including id;

On the other hand, if there is ρ'(p, yi) = p' and p' ∈κ'(p, xi), since the path is unique for id, therefore they can only be computed by [id[($\bar{x}$)< $\bar{y}$ >]], and since σ' is effective, therefore, yi is bound by an imported name of path p' in a and xi is binding an exported name of path p' in a.

Proposition 2. Assume that id[($\bar{x}$) [μ1 …μr] < $\bar{y}$ >] is one subaction of an action a, p is its path in a, σ is the right set of names and their paths that is effective for id, if [id[($\bar{x}$) [μ1 …μr]

< $\bar{y}$ >]]pσρκ= (σ', ρ' , κ') , and if [μi](cons(p,id(μi)), σi-1, ρi-1 , κi-1) are correct, then we have :

(1) y_i is bound by an imported name of path p' in *a* iff ρ'(p, y_i) = p' ;
(2) xi is binding an exported name of path p' in a iff p' ∈ κ'(p, xi).

Proof: From the definition of ⟦ id[($\bar{x}$) [μ_1 …μ_r]< $\bar{y}$ >]⟧ we can see that:

(1) σ' is just obtained by updating σ with current new effective imported names x; since σ is right, therefore σ' is right too;
(2) we can sequentially compute binding and bound environment for every molecule, and we know that [μi](cons(p,id(μi)), σi-1, ρi-1 , κi-1) are correct; then for exported names of id, the computation is processed with the result of all the molecules, which is also correct. Therefore, adding the correct result of molecules and that of the exported names of id to original (ρ, κ) will produce a correct one so far including id; On the other hand, if there is ρ'(p, yi) = p' and p' ∈κ'(p, xi), since the path is unique for id, therefore they can only be computed by [id[($\bar{x}$) [μ1 …μr] < $\bar{y}$ >]], and since path environment is effective, therefore, yi is bound by an imported name of path p' in a and xi is binding an exported name of path p' in a.

Proposition 3. Assume that id[< $\bar{u}$ > K ($\bar{v}$)] is a molecule of one subaction of an action *a*, p is its path in *a*, σ is the right set of names and their paths that is effective for id, if ⟦id[< $\bar{u}$ > K ($\bar{v}$)]⟧pσρκ= (σ', ρ' , κ') then we have :

(1) u_i is bound by an imported name of path p' in *a* iff ρ'(p, u_i) = p' ;
(2) vi is binding an exported name of path p' in a iff p' ∈κ'(p, vi).

Proof: The proof is similar to preposition 1.

Proposition 4. Assume that id[< $\bar{u}$ > Ka ($\bar{v}$)] is a molecule of one subaction of an action *a*, p is its path in *a*, σ is the right set of names and their paths that is effective for id, if ⟦id[< $\bar{u}$ > Ka ($\bar{v}$)]⟧pσρκ= (σ', ρ' , κ') , and if ⟦a_i⟧(cons(p,id(a_1)), σ', Φ, Φ) are correct, then we have :

(1) u_i is bound by an imported name of path p' in *a* iff ρ'(p, u_i) = p' ;
(2) vi is binding an exported name of path p' in a iff p' ∈κ'(p, vi).

Proof: The proof is similar to preposition 2.

Lemma 1. For an action *a* whose identifier is id_a, we assume that the initial abstract environment is { $\sigma_0 = [x \rightarrow ()]$; $\rho_0 = [(p, x) \rightarrow ()]$; $\kappa_0 = [(p, x) \rightarrow \{\}]$ }, for any name x occurring in *a*, and the initial path p is (id_a). If $[\![a]\!]p\sigma_0\rho_0\kappa_0 = (\sigma, \rho, \kappa)$, then we have:

(1) an exported name x of path p_1 is bound by the imported name of path p_2 in *a* iff $\rho(p_1, x) = p_2$;

(2) an imported name x of path p1 is binding an exported name of path p2 in a iff $p2 \in \kappa(p1, x)$.

Proof:

(1) for any action, the initial abstract given here as input is no doubt correct;

(2) since the number of subactions of an action is finite number, therefore, the [a]pσ0ρ0κ0 will surely terminate;

(3) for any action a, from the definition of the analyzing function, computation of [a]pσ0ρ0κ0 will be processed recursively until meeting the constant action or a molecule with constant control, from the preposition 1, 2, 3, 4 we can prove that since in every step the path and the path environment are correct, therefore in every step we result in exact binding environment and bound environment, so in the termination we have the exact binding environment and bound environment for that action, which satisfy (1) and (2).

4 Security Properties

In this section we will present the security policies for both static and dynamic action calculi in terms of corresponding security properties, and then establish the relationship between the static notation and dynamic one.

We use the result of the above formal flow analysis to establish security properties mainly integrating the idea of [1, 3], which is to ensure that an action preserve the secrecy of data.

Similar to [3], we partition names with its paths defined in an action a into trustworthy T(a) and untrustworthy U(a). We say that the secrecy of data is preserved if an action a never read an untrustworthy name to it and an action a never write to other actions the untrustworthy name, which can be represented by following formal definition.

Definition 8. A subaction *a* in action *b* with occurrence path p is *defended*, iff $(\sigma, \rho, \kappa) = [\![b]\!]p_0\sigma_0\rho_0\kappa_0$, and there is no such $(x, p_x) \in U(a)$ that $p \in \kappa(x, p_x)$, and $\forall x \in exp(a)$, $(x, \rho(x,p)) \in T(a)$.

Definition 9. A subaction *a* in action *b* with occurrence path p has *no leakage* iff $(\sigma, \rho, \kappa) = [\![b]\!]p_0\sigma_0\rho_0\kappa_0$, and there is no such *c* of path p' in *b*, $Path(c, a) \cap Path(b, a) \neq \Phi$, that $\forall x \in (U(c) \cap names(c))$, $\sigma(x, p') = p$, and $\forall x \in imp(a)$, $p'' \in \kappa(x, p)$, $(x, p'') \in T(d)$, where $Path(d, a) = p''$.

Definition 10. An action *a* preserves secrecy of data iff all the subactions of *a* are defended and have no leakage.

The definitions above are static security notations for action calculi. Next we would like to present dynamic secrecy notation for action calculi. Since the dynamics need

to be defined when you are defining a concrete action calculus, the concrete form of the reactive rule are undefined, therefore, we can only give a general result.

Definition 11. An action a is *protected* iff whenever $a \searrow b$, and c is communicating y to d via x in this reaction, we have $x \in T(c)$ and $x \in T(d)$, $y \in T(d)$, where x is a name, c and d are subaction of a,

Lemma 2. If an action a preserves secrecy of data then it is protected.
Proof: From the definition 8, 9 and 10, if an action a preserves secrecy of data, then we have that every subaction b of a are defended and have no leakage, which means:

$\forall x \in exp(c), (x, \rho(x, p)) \in T(c)$.
$\forall x \in exp(d), (x, \rho(x, p)) \in T(d)$.
$\forall x \in imp(a), p'' \in \kappa(x, p), (x, p'') \in T(d)$, where $Path(d, a)=p''$
where $(\sigma, \rho, \kappa)=[\![b]\!]p_0\sigma_0\rho_0\kappa_0$.

x is a name, if $a \searrow b$, c, d are subaction of a, and c is communicating y to d via x in this reaction, then we have:

(1) x must be exported names of c and d;
(2) y must be imported name of c and exported name of d satisfying that y in c is binding y in d;

therefore, we can conclude that $x \in T(c)$ and $x \in T(d)$, $y \in T(d)$. So according to definition 11 we can say that a is protected.

5 Related Work and Conclusion

Applying static analysis of formal models in studying security properties of concurrent systems is an active and interesting research topic in formal methods in recent years[1,2].There is a large body of research on information flow control aiming at specifying, verifying and analyzing security.

Secure information flow in programming language received its recent reincarnation. Many researchers have investigated the problem including [12,13,14,15,16], which all use concrete calculi or programming languages as research target, while this paper use action calculi, an abstract model for a class of calculi.

Recently, the use of type systems for information flow is also developed[3, 9, 16].

In [1, 2, 3, 4] data flow or control flow analysis for pi calculus, safe ambients, CML have been presented, and some encouraging results in proving security property are established.

The idea of using information flow analysis techniques for studying the security properties of action calculi arises from that of pi calculus and ambient[1, 2, 3]. We propose information flow analysis for static action calculi, and use it for the validation of rather simple security properties of interactive systems. More complicated security properties will be studied in the future. Because action calculi is a framework for a class of models for interactive system, these work can open a way to study the generality of these properties by comparing the results with those of each concrete model for interactive system.

References

1. C. Bodel, P. Deggan, F. Nielson, and H. Riis Nielson. Static Analysis for the π-calculus with Applications to Security. Available at http://www.di.unipi.it/~chiars/publ-40/BDNN100.ps.
2. C. Bodei, P. Degano, F. Nielson, and H. Riis Nielson. Control flow analysis for the π-calculus. In Proceedings of CONCUR'98, LNCS 1466, pp84-89. Springer-Verlag, 1998.
3. P. Degano, F. Levi and C. Bodei. Safe Ambients: Control Flow Analysis and Security. In Proceedings of ASIAN '00. LNCS 1961, 199-214. Springer-Verlag.
4. Gasser, K.L.S., Nielson, F., Nielson H.R. Systematic realization of control flow analysis for CML. In Proceedings of ICFP'97, P38-51. ACM press, 1997.
5. Gardner, P.. Closed Action Calculi. Theoretical Computer Science, vol. 228, 1999.
6. Milner, R.. "Action calculi, or concrete action structures", Proc. MFCS Conference, Gdansk, Poland, LNCS, Vol. 711, Springer-Verlag, pp105-121, 1993.
7. Milner R.. Action Calculi IV: Molecular forms, Computer Sciences Department, University of Edinburgh. Draft(1993).
8. Milner R.. " Calculi for Interaction", Acta Inform. 33(8), pp 707-737, 1996.
9. Silvia Crafa, Michele Bugliesi, Giuseppe Castagna. Information Flow Security in Boxed Ambients. Electronic Notes in Theoretical Computer Science 66, No.3(2003.
10. Ying JIN, Chengzhi JIN. Encoding Gamma Calculus in Action Calculus. Journal of Software, Vol. 14, No. 1, January 2003.
11. Ying JIN, Chengzhi JIN. Representing Imperative Language in Higher-order Action Calculus. Journal of Computer Research and Development, Vol. 39, No.10, Oct. 2002.
12. R. Focardi. Analysis and Automatic Detection of Information Flows in Systems and Networks. PhD thesis, Unviersity of Bologna(Italy), 1999.
13. R. Focardi, R. Gorrieri, and F. Martinelli. Information Flow Analysis in a Discrtete Time Process Algebra. In Proceedings of 13th IEEE Computer Security Foundations Workshop(CSFW13), P. Syverson ed., Page 170-184. IEEE CS Press, July 2000.
14. R. Focardi, R. Gorrieri, and R. Segala. A New Definition of Security Properties. In Proceedings of Workshop on Issues in the Theory of Security(WITS'00), University of Geneva, July 2000.
15. Heiko Mantel, Andrei Sabelfeld. A Unifying Approach to Security of Distributed and Multi-Threaded Programs. In Proceedings of the 14th IEEE Computer Security Foundations Workshop, Cape Breton, Nova Scotia, Canada, June 2001.
16. Andrei Sabelfeld, Andrew C. Myers. Language-Based Information-Flow Security. IEEE Journal of Selected Areas in Commnications, Vol. 21, No. 1, January 2003.

A Microeconomics-Based Fuzzy QoS Unicast Routing Scheme in NGI*

Xingwei Wang, Meijia Hou, Junwei Wang, and Min Huang

College of Information Science and Engineering, Northeastern University, Shenyang, 110004, China
wangxw@mail.neu.edu.cn

Abstract. Due to the difficulty on exact measurement and expression of NGI (Next-Generation Internet) network status, the necessary QoS routing information is fuzzy. With the gradual commercialization of network operation, paying for network usage calls for QoS pricing and accounting. In this paper, a microeconomics-based fuzzy QoS unicast routing scheme is proposed, consisting of three phases: edge evaluation, game analysis, and route selection. It attempts to make both network provider and user utilities maximized along the found route, with not only the user QoS requirements satisfied but also the Pareto-optimum under the Nash equilibrium on their utilities achieved.

1 Introduction

Recently, with the convergence of Internet, multimedia content, and mobile communication technology, the NGI (Next-Generation Internet) is becoming an integrated network, including terrestrial-based, space-based, sky-based, fixed and mobile subnetworks, supporting anywhere, anytime with any kind of information to communicate with anyone or even any object in fixed or mobile way [1]. In order to provide the user with the end-to-end QoS (Quality of Service) support, each part of NGI should support the QoS, with wired QoS and wireless QoS converged seamlessly. However, it is hard to describe the network status exactly and completely. With gradual commercialization of the network operation, paying for network usage become necessary, QoS pricing and accounting should be provided [2]. However, the network providers pursue profit as much as possible, while the users wish to get the best service with the smallest cost. There exist conflicts on profits between the network providers and their users, and thus "both-win" should be attained. Support from QoS routing should be provided to help the above problems solved.

Although a lot of researches have been done on the QoS routing, the simultaneous network provider and user utility optimization and network status fuzziness are not considered in depth [3-6]. In this paper, a microeconomics-based fuzzy QoS unicast routing scheme is proposed. It attempts to make both the network provider and the user utilities maximized along the found route, and no other routes can make the two

* This work is supported by the National Natural Science Foundation of China under Grant No. 60473089, No. 60003006 and No. 70101006; the Natural Science Foundation of Liaoning Province under Grant No. 20032018 and No. 20032019; Modern Distance Education Engineering Project by China MoE.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 1055 – 1064, 2005.
© IFIP International Federation for Information Processing 2005

utilities improved simultaneously any more unless one of them is sacrificed. Thus, the Pareto-optimum under the Nash equilibrium [7] on both the network provider and the user utilities is achieved or approached along the found route.

2 Problem Formulations

A network can be modeled as a graph $G(V,E)$, where V is the set of nodes representing routers and E is the set of edges representing links. For each node $v_j \in V(j=1,2,3,\ldots,|V|)$, consider the following parameters: delay, delay jitter, and error rate; for each edge $e_{jk} \in E(j,k=1,2,3,\ldots,|V|)$ between $\forall v_j, v_k \in V$, consider the following parameters: available bandwidth, delay and error rate. Just for the algorithm description simplicity, the parameters of node are merged into those of its upstream edge along the route. Thus, the parameters of the edge become as follows: available bandwidth bw_{jk}, delay del_{jk}, delay jitter jt_{jk}, and error rate ls_{jk}.

Suppose that the source node is $v_s \in V$ and the destination node is $v_t \in V$, look for the specific route p_{st} between v_s and v_t, trying to make the network provider utility TNU and the user utility TUU achieve or approach the Pareto-optimum under the Nash equilibrium as much as possible with the following constraints satisfied:

A1) The available bottleneck bandwidth along p_{st} is not smaller than the user bandwidth requirement bw_req.

A2) The delay along p_{st} is not bigger than the user delay requirement del_req.

A3) The delay jitter along p_{st} is not bigger than the user delay jitter requirement jt_req.

A4) The error rate along p_{st} is not bigger than the user error rate requirement ls_req.

The corresponding mathematical model is described as follows:

$$TNU + TUU \rightarrow \max\{TNU + TUU\} \tag{1}$$

$$TNU \rightarrow \max\{TNU\} \tag{2}$$

$$TUU \rightarrow \max\{TUU\} \tag{3}$$

$$\min\{bw_{jk} \mid e_{jk} \in p_{st}\} \geq bw_req \tag{4}$$

$$\sum_{e_{jk} \in p_{st}} del_{jk} \leq del_req \tag{5}$$

$$\sum_{e_{jk} \in p_{st}} jt_{jk} \leq jt_req \tag{6}$$

$$1-\prod_{e_{jk}\in p_{st}}\left(1-ls_{jk}\right)\le ls_req \tag{7}$$

Among them, $TNU=\sum_{e_{jk}\in p_{st}} nu_{jk}$, $TUU=\sum_{e_{jk}\in p_{st}} uu_{jk}$, nu_{jk} and uu_{jk} represent the network provider utility and the user utility on the edge e_{jk} respectively.

The above problem is NPC [8], and is resolved by the proposed scheme.

3 QoS Routing Scheme Descriptions

The proposed QoS routing scheme in this paper consists of three parts: edge evaluation, game analysis and route selection.

3.1 Edge Evaluation

The adaptability membership degree function is used to describe the adaptability of the candidate edge conditions to the user QoS requirements and is defined as follows:

$$g_1(bw,bw_req)=\begin{cases}0 & bw<bw_req\\ \left(\dfrac{bw\text{-}bw_req}{b\text{-}bw_req}\right)^q+f_1(bw,bw_req) & bw_req\le bw<b\\ 1 & bw\ge b\end{cases} \tag{8}$$

Among it, $f_1(bw,bw_req)=\begin{cases}0 & otherwise\\ \varepsilon & bw=bw_req\end{cases}$.

The edge delay adaptability membership degree function is defined as follows:

$$g_2(Jp,del,del_req)=\begin{cases}0 & del>del_req\\ 1-e^{-\left(\frac{del_req-del}{\sigma_1}\right)^2}+f_2(Jp,del,del_req) & del\le del_req\end{cases} \tag{9}$$

Among it, $f_2(Jp,del,del_req)=\begin{cases}0 & otherwise\\ \varepsilon & Jp=1\wedge del=del_req\end{cases}$.

The edge delay jitter adaptability membership degree function is defined as follows:

$$g_3(Jp,jt,jt_req)=\begin{cases}0 & jt>jt_req\\ 1-e^{-\left(\frac{jt_req\text{-}jt}{\sigma_2}\right)^2}+f_3(Jp,jt,jt_req) & jt\le jt_req\end{cases} \tag{10}$$

Among it, $f_3(Jp,jt,jt_req)=\begin{cases}0 & otherwise\\ \varepsilon & Jp=1\wedge jt=jt_req\end{cases}$.

The edge error rate adaptability membership degree function is defined as follows:

$$g_4(Jp, ls, ls_req) = \begin{cases} 0 & ls > ls_req \\ 1 - e^{-\left(\frac{ls_req - ls}{\sigma_3}\right)^2} + f_4(Jp, ls, ls_req) & ls \le ls_req \end{cases} \tag{11}$$

Among it, $f_4(Jp, ls, ls_req) = \begin{cases} 0 & otherwise \\ \varepsilon & Jp = 1 \wedge ls = ls_req \end{cases}$.

(9)-(11) all are Gaussian alike with smooth transition feature. $f_h(h = 1,2,3,4)$ is used to deal with the special one hop route case. Jp is a positive integer, representing the hop count of end-to-end route. ε is a positive pure decimal fraction and much smaller than 1. bw, del, jt and ls are the available bandwidth, delay, delay jitter and error rate of the candidate edge respectively. q, b, σ_1, σ_2 and σ_3 all are positive constants, $q > 1$. An evaluation matrix $R = [g_1, g_2, g_3, g_4]^T$ of the candidate edge can be gotten from (8)-(11). According to the application nature, a weight matrix $W = [w_1, w_2, w_3, w_4]$ $(0 < w_1, w_2, w_3, w_4 < 1)$ is given. Here, w_1, w_2, w_3 and w_4 are the significance weights of bandwidth, delay, delay jitter and error rate on the application QoS respectively. The comprehensive evaluation value ω on the candidate edge conditions with regard to the user QoS requirements is computed as follows:

$$\omega = W \times R \tag{12}$$

The bigger the value of ω is, the higher the adaptability of the candidate edge conditions to the user QoS requirements is.

Whether the available bandwidth of the candidate edge is abundant can be derived from the result of (8), and thus the bandwidth supply and demand relation of the candidate edge can be deduced. If $g_1 < h_1$ (h_1 is a constant and $0 < h_1 < 1$), the available bandwidth of the candidate edge is considered to be scarce; if $h_1 \le g_1 < h_2$ (h_2 is a constant and $0 < h_1 < h_2 < 1$), the available bandwidth of the candidate edge is considered to be moderate; if $g_1 \ge h_2$, the available bandwidth of the candidate edge is considered to be abundant. Thus, a tuning coefficient for the amount of bandwidth to be actually allocated to the user is introduced, and its definition is as follows:

$$\rho = \begin{cases} \rho_1 & g_1 < h_1 \\ \rho_2 & h_1 \le g_1 < h_2 \\ \rho_3 & g_1 \ge h_2 \end{cases} \tag{13}$$

Among it, $0 < \rho_1 < 1$, $\rho_2 = 1$, $\rho_3 > 1$, the values of ρ_1 and ρ_3 are preset according to the actual experience. The actually allocated amount of bandwidth nbw to the user on the candidate edge is calculated as follows:

$$nbw = \rho * bw_req \tag{14}$$

3.2 Game Analysis

In this paper, there are two players in the game, that is, the network provider and the user. The network provider has two game strategies: s_1 and s_2, denoting whether it is willing to provide the bandwidth of the candidate edge to the user or not respectively; the user also has two game strategies: t_1 and t_2, denoting whether he is willing to accept the provided bandwidth of the candidate edge respectively.

The network provider and the user game matrixes, NM and UM, are defined as follows:

$$NM = \begin{bmatrix} pn_{11} & pn_{12} \\ pn_{21} & pn_{22} \end{bmatrix} \tag{15}$$

$$UM = \begin{bmatrix} pu_{11} & pu_{12} \\ pu_{21} & pu_{22} \end{bmatrix} \tag{16}$$

Here, the rows in NM and UM correspond to the game strategies s_1 and s_2 of the network provider, and the columns correspond to the game strategies t_1 and t_2 of the user. The element $pn_{mn}(m,n=1,2)$ in NM represents the relative utility of the network provider on the candidate edge for s_m and t_n; the element $pu_{mn}(m,n=1,2)$ in NM represents the relative utility of the user on the candidate edge for s_m and t_n.

After the edge evaluation described in section 3.1, the comprehensive evaluation value of ω on the candidate edge has been gotten. According to the actual experience, a threshold value ω_0 is set. If $\omega > \omega_0$, the actual status of the candidate edge is considered better than that the user expected; if $\omega = \omega_0$, the actual status of the candidate edge is considered to be just what the user expected; if $\omega < \omega_0$, the actual status of the candidate edge is considered worse than that the user expected. Therefore, the matrix element values of NM and UM are given as follows:

$$NM = \begin{bmatrix} \dfrac{\left(uct\dfrac{\omega}{\omega_0} - uct\right)}{nbw} & \dfrac{\left(uct\dfrac{\omega}{\omega_0} - uct\right)}{nbw} \\ -\mu\dfrac{\left(uct\dfrac{\omega}{\omega_0} - uct\right)}{nbw} & -\dfrac{\left(uct\dfrac{\omega}{\omega_0} - uct\right)}{nbw} \end{bmatrix} \tag{17}$$

$$UM = \begin{bmatrix} \dfrac{nbw\dfrac{\omega}{\omega_0}}{uct} - \dfrac{nbw}{uct} & -\mu\left(\dfrac{nbw\dfrac{\omega}{\omega_0}}{uct} - \dfrac{nbw}{uct}\right) \\ \dfrac{nbw\dfrac{\omega}{\omega_0}}{uct} - \dfrac{nbw}{uct} & -\left(\dfrac{nbw\dfrac{\omega}{\omega_0}}{uct} - \dfrac{nbw}{uct}\right) \end{bmatrix} \tag{18}$$

Among (17) and (18), uct denotes the amount of money that the user should pay for his usage of the candidate edge. In NM, $uct\frac{\omega}{\omega_0}/nbw$ represents the virtual utility of the network provider on the candidate edge, and uct/nbw represents its actual utility, the difference between which represents the relative utility of the network provider on the candidate edge. The minus sign in pn_{21} and pn_{22} mean that, if the network provider rejectes the user, its utility will be lost. μ is a penalty factor and its value is set bigger than 1 [9], meaning that rejecting one willing user would bring the negative effect on this and other users' willingness to use the services provided by the network provider in the future considerably. Similarly, in UM, $nbw/uct\frac{\omega}{\omega_0}$ represents the virtual utility of the user on the candidate edge, and nbw/uct represents his actual utility, the difference between which represents the relative utility of the user on the candidate edge. The negative values of elements and μ in UM have the similar meanings as those in NM. In NM or UM, if the values of pn_{mn} and pu_{mn} are negative, it does mean that the network provider and/or the user are/is not satisfied on the current game strategy combinations. If the following inequations [10] are satisfied:

$$\begin{cases} pn_{m^*n^*} \geq pn_{mn^*} \\ pu_{m^*n^*} \geq pu_{m^*n} \end{cases} \qquad m, n = 1,\ 2 \tag{19}$$

The corresponding strategy pair $\{s_{m^*}, t_{n^*}\}$ represents a pair of non-cooperative pure strategies, namely the specific solution under Nash equilibrium [11], here, m^* and n^* stand for some m and n.

3.3 Route Selection

Heuristic cost. After the game result of the candidate edge e_{jk} is gotten, it is transformed into one kind of weight, denoted by Ω_{jk}, which is defined as follows:

$$\Omega_{jk} = \begin{cases} 1 & Nash\ \ Equilibrium \\ >1 & non\text{-}Nash\ \ Equilibrium \end{cases} \tag{20}$$

The heuristic cost $T_{f_{jk}}\left(\Omega_{jk}, nu_{jk}, uu_{jk}\right)$ of e_{jk} is defined as follows:

$$T_{f_{jk}}\left(\Omega_{jk}, nu_{jk}, uu_{jk}\right) = \Omega_{jk}\left(q_1 \frac{1}{nu_{jk}} + q_2 \frac{1}{uu_{jk}}\right) \tag{21}$$

In formula (21), Ω_{jk} represents the influence of Nash equilibrium on the route selection. q_1 and q_2 are the preference weights, representing whether and how much

the network provider/user utility should be considered with priority when routing. nu_{jk} and uu_{jk} use the actual utility of the network and the user respectively.

The objective of the proposed scheme in this paper is to minimize the heuristic cost sum along the route, that is,

$$minimize\left\{\sum_{e_{jk}\in p_{st}} T_{f_{jk}}\left(\Omega_{jk}, nu_{jk}, uu_{jk}\right)\right\} \tag{22}$$

Routing Algorithm. v_s and v_t are source and destination node respectively. Let pc and Tc denote pc label and Tc label of node v. $pc(v)$ is the minimum heuristic cost from v_s to v with the specific constraints satisfied. $Tc(v)$ is the upper bound of $pc(v)$. S_i is the set of those nodes with pc label at Step i. Each node is given a λ. When the proposed algorithm ended, if $\lambda(v)=m$, the precedent node of v along the route with the minimum heuristic cost is v_m; if $\lambda(v)=m'$, there does not exist the satisfied route from v_s to v; if $\lambda(v)=0$, $v=v_s$. How to assign certain value to λ is described in (6) of 2*nd* labeling at Step1, that is, when the 1*st* and 2*nd* labeling conditions are met with, the λ value of the considered node is marked as the number of the specific node leading to it along the route with the minimum heuristic cost. In addition, $minbw(v_j)$ is the available bottleneck bandwidth; $del(v_j)$ is the delay; $jt(v_j)$ is the delay jitter; and $ls(v_j)$ is the error rate along the path from v_j to v_s. $T_{f_{kj}}$, bw_{kj}, del_{kj}, jt_{kj}, and ls_{kj} are the heuristic cost, the available bandwidth, the delay, the delay jitter, and the error rate of the edge e_{kj} respectively. Based on the algorithm proposed in [8], the following routing algorithm is designed:

Step0. Initialization: $i=0$, $S_o=\{v_s\}$, $\lambda(v_s)=0$. $\forall v\neq v_s$, $Tc(v)=+\infty$, $\lambda(v)=m'$, $k=s$.

(1) $pc(v_k)=0$;

(2) $minbw(v_k)=+\infty$;

(3) $del(v_k)=0$, $jt(v_k)=0$, $ls(v_k)=0$;

Step1. Labeling procedure

For each node v_j with $e_{kj}\in E$ and $v_j\notin S_i$, compute $T_{f_{kj}}$ according to (8)-(21).

1*st* labeling condition: in order to meet with the objective of (22), if $Tc(v_j)>pc(v_k)+T_{f_{kj}}$, compute as follows:

(1) $pc'(v_j)=pc(v_j)+T_{f_{kj}}$;

(2) $minbw'(v_j)=min\{minbw(v_k), bw_{kj}\}$;

(3) $del'(v_j)=del(v_k)+del_{kj}$, $jt'(v_j)=jt(v_k)+jt_{kj}$, $ls'(v_j)=1-(1-ls(v_k))(1-ls_{kj})$;

2*nd* labeling condition: according to (4)- (7), if

(1) $min\,bw'(v_j) \geq bw_req$;

(2) $del'(v_j) \leq del_req$, $jt'(v_j) \leq jt_req$, $ls'(v_j) \leq ls_req$;

then

(1) $Tc(v_j) = pc'(v_j)$;

(2) $min\,bw(v_j) = min\,bw'(v_j)$;

(3) $del(v_j) = del'(v_j)$, $jt(v_j) = jt'(v_j)$, $ls(v_j) = ls'(v_j)$;

(4) $\lambda(v_j) = k$;

go to Step2; otherwise, negotiate with user: if succeeded, go to Step2, otherwise the algorithm ends.

Step2. Modification procedure

In order to meet with the objective of (22), let $H_1 = \left\{ v_{j_i} \mid \min\limits_{v_{j_i} \notin S_i} \left\{ Tc(v_{j_i}) \right\} \right\}$. For any $v_{j_i} \in H_1$, if $Tc(v_{j_i}) < +\infty$, go to Step2.1; otherwise, there does not exist any feasible solution, and then negotiate with the user: if succeeded, go to Step2.6, otherwise the algorithm ends.

Step2.1. If $|H_1| = 1$, get $v_{j_i} \in H_1$, and go to Step2.6; otherwise go to Step2.2.

Step2.2. Let $H_2 = \left\{ v_{j_i} \mid \max\limits_{v_{j_i} \in H_1} \left\{ \min bw(v_{j_i}) - bw_req \right\} \right\}$. If $|H_2| = 1$, get $v_{j_i} \in H_2$, and go to Step2.6; otherwise go to Step2.3.

Step2.3. Let $H_3 = \left\{ v_{j_i} \mid \max\limits_{v_{j_i} \in H_2} \left\{ del_req - del(v_{j_i}) \right\} \right\}$. If $|H_3| = 1$, get $v_{j_i} \in H_3$, and go to Step2.6; otherwise go to Step2.4.

Step2.4. Let $H4 = \left\{ v_{j_i} \mid \max\limits_{v_{j_i} \in H_3} \left\{ jt_req - jt(v_{j_i}) \right\} \right\}$. If $|H_4| = 1$, get $v_{j_i} \in H_4$, and go to Step2.6; otherwise go to Step2.5.

Step2.5. Let $H_5 = \left\{ v_{j_i} \mid \min\limits_{v_{j_i} \in H_4} \left\{ ls(v_{j_i}) \right\} \right\}$. If $|H_5| = 1$, get $v_{j_i} \in H_5$, and go to Step2.6; otherwise, get any $v_{j_i} \in H_5$, and go to Step2.6.

Step2.6. Modify the Tc label to pc label, that is, let $pc(v_{j_i}) = Tc(v_{j_i})$ and $S_{i+1} = S_i \cup (v_{j_i})$, $k = j_i$, $i = i + 1$, if $k = t$, output the results and the algorithm ends, otherwise go to Step1.

4 Performance Evaluations and Conclusions

Simulations have been done on NS (Network Simulator) 2 platforms [12]. SPF-based unicast routing scheme, fuzzy-tower-based QoS unicast routing scheme and the scheme proposed in this paper have been performed over some actual and virtual

network topologies (Fig.1, Fig.2 and Fig.3 are three examples), and performance comparisons among them have been done. For simplicity, the above three schemes are denoted by SPF, FTQ, and MFQ respectively.

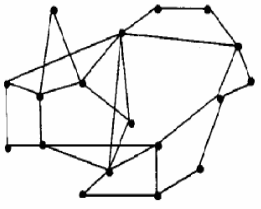


Fig. 1. The 1*st* topology

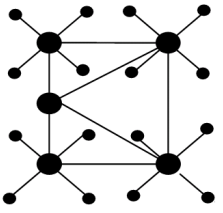


Fig. 2. The 2*nd* topology

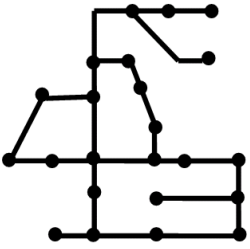


Fig. 3. The 3*rd* topology

About the network provider utility, the user utility and the comprehensive utility (the network provider utility plus the user utility), SPF:MFQ:FTQ over 1*st*, 2*nd* and 3*rd* topologies are shown in Fig.4, Fig.5 and Fig.6 respectively. Simulation results have shown that the proposed scheme is effective and efficient.

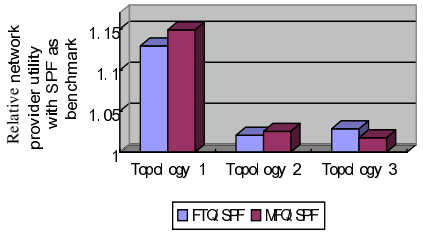


Fig. 4. Comparison of network provider utility

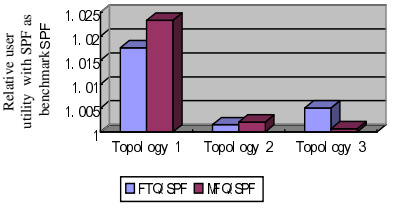


Fig. 5. Comparison of user utility

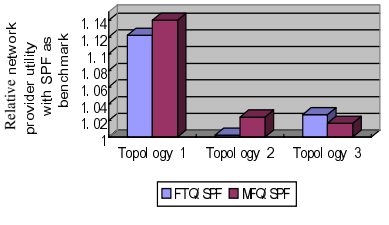


Fig. 6. Comparison of comprehensiveutility

In future, the proposed scheme will be improved on its practicability with its prototype systems developed and its extensions to multicast scenarios will also be done. In addition, taking the difficulty on exact and complete expression of the user QoS requirements into account, how to tackle the fuzziness of both the user QoS requirements and the network status in our proposed scheme is another emphasis of our future research.

References

1. Keiji, T.: A Perspective on the Evolution of Mobile Communications. IEEE Communications Magazine, Vol. 41, No.10. (2003)66-73
2. Theodore, B. Z., Konstantinos, G. V., Christos, P. T., et al: Global Roaming in Next-Generation Networks. IEEE Communications Magazine, Vol. 40, No.2. (2002)145-151
3. Douglas, S., Hussein, F.: A Distributed Algorithm for Delay-Constrained Unicast Routing. IEEE/ACM Transactions on Networking, Vol. 8, No. 2. (2000)239-250
4. Wang, Z., Crowcroft, J.: QoS Routing for Supporting Resource Reservation. IEEE Journal on Selected Areas in Communications, Vol. 14, No.7. (1996)1228-1234
5. Wang, X. W., Yuan, C. Q., Huang, M.: A Fuzzy-tower-based QoS Unicast Routing Algorithm[A]. Proc. of Embedded And Ubiquitous Computing(EUC'04)[C], Aizu: Springer LNCS 3207, 923-930, (2004)
6. Zhao, J., Wu, J. Y., Gu, G. Q.: A Class of Network Quality of Service Based Unicast Routing Algorithm. Journal of China Institute of Communications, Vol. 22, No. 11. (2001) 30-41
7. Quan, X. T., Zhang, J.: Theory of Economics Game. Beijing: China Machine Press, (2003)
8. Wang, X. W., Wang, Z. J., Huang, M, et al: Quality of Service Based Initial Route Setup Algorithms for Multimedia Communication. Chinese Journal of Computers, Vol. 24, No. 8. (2001)830-837
9. Marco, D.: Ant Algorithms Solve Difficult Optimization Problems. Advances in Artificial Life : 6th European Conference, Prague: Czech Republic, (2001)11-22
10. Yuan, X., Liu, X.: Heuristic Algorithm for Multi-constrained Quality of Service Routing Problem. Proceeding of The IEEE Infocom 2001, Piscataway: IEEE Communication Society, (2001)844-853
11. Shi, X. Q.: Game Theory. Shanghai: Shang Hai University of Finance & Economics Press, (2000)
12. Xu, L. M., Pang, B., Zhao, R.: NS and network simulation. Beijing: Posts & Telecom Press, (2003)

Considerations of Point-to-Multipoint QoS Based Route Optimization Using PCEMP

Dipnarayan Guha[1], Seng Kyoun Jo[1], Doan Huy Cuong[1], and Jun Kyun Choi[2]

[1] Researcher, BcN ITRC, Broadband Network Laboratory,
Information and Communications University,
119 Munji-Dong, Yuseong-Gu, Daejeon 305-714, Republic of Korea
{dip, skjo, cuongdh}@icu.ac.kr
[2] Associate Professor, BcN ITRC, Broadband Network Laboratory,
Information and Communications University,
119 Munji-Dong, Yuseong-Gu, Daejeon 305-714, Republic of Korea
jkchoi@icu.ac.kr

Abstract. This paper describes the basic concepts of point-to-multipoint (p2mp) path computation on the basis of the Path Computation Element Metric Protocol (PCEMP). PCEMP, being soft-memory based, has the capability of dynamic configuration of its finite state machines (FSMs) in the participating PCEMP peers, and thus can support a wide variety of traffic engineering techniques that are needed to guarantee bandwidth demand and scalable fast protection and restoration in PCE based p2mp frameworks ensuring end-to-end QoS support. The authors have proposed this concept in the newly constituted PCE WG (Path Computation Element Work Group) in the RTG sub-area of the IETF. In this research-in-progress paper, we show how PCEMP as it is defined, and the optimal number of PCE Domain Areas (PCEDAs) that might be allocated to a PCE node for the best performance in end-to-end QoS management based on a tight optimal Cramer Rao bound for the state machine executions.

1 Introduction

One of the key work items involving the functional specification of MPLS and GMPLS Traffic Engineering LSP path computation techniques in the proposed PCE WG [1] charter is the case for TE LSP path computation for inter-domain areas applying to both point-to-point (p2p) and point-to-multipoint (p2mp) TE LSPs. Most of the existing MPLS TE allows for strict QoS guarantee, resource optimization and fast failure recovery, but the scope is mostly limited to p2p applications [2]. In the context of path computation, one of the important application areas is the reliable support of bandwidth-on-demand applications, where the QoS provisioning needs to be dynamic and robust. A scenario where a PCE node acts a server which are connected to several clients, which may or may not be PCE peers, needs a clear requirement addressal so far as p2mp TE tunneling is considered. In this paper, we consider that such p2mp TE LSP path computation is QoS triggered, and we show how PCEMP finite state machines (FSMs) might help in achieving a scalable architecture involving PCEDAs where p2mp path computation metrics are independent of the number of clients to which the PCE server is attached to. The Path Computation Element Metric Protocol

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 1065 – 1074, 2005.
© IFIP International Federation for Information Processing 2005

(PCEMP) [3] acts as a generic computational model for path based metrics in large multi-domain and/or multi-layer networks. This paper also shows that feature of the PCEMP protocol in degenerating the setup and teardown of p2mp TE LSP computation to the PCEMP protocol processing itself, thus enabling support of an arbitrary number of clients as well provisioning of guaranteed robust path protection and restoration and dynamic QoS provisioning for bandwidth-on-demand services [4].

2 p2mp QoS Based Path Computation Fundamentals

For the scenario involving robust and dynamic provisioning of bandwidth-on-demand services, the p2mp applications request p2mp forwarding paths in case of different topology deployments. The robustness must be thought in the context of path re-optimization, so a quick change in the topology must be accommodated with every PCEDA level optimization. The p2mp path will have several observed metrics as constraints, such as cost of path establishment and teardown, delay bounds of the p2mp path, both delay bounded and cost optimized constraints in tandem for path computation, etc. One of the features as brought out in the PCE WG charter is the co-existence of different path computation algorithms on the PCE node, so that depending upon the data that is processed, a particular algorithm is invoked. It is also evident that for p2mp applications, a CPU intensive path computation is necessary, primarily because most of the bandwidth-on-demand applications tend to be resource-intensive applications like streaming multimedia, real-time videoconferencing, etc. The ideal thing would be to let the data that is under processing in the PCE node determine the path computation algorithm directly, which would mean that the constraints imposed by the QoS provisioning requirements would directly determine the path computation algorithm and path re-optimization, which in turns drives the resulting topology architecture. Thus, it is easy to see why PCEMP is a possible solution for p2mp TE LSP computation, as it drives a protocol driven architecture for topology changes in path re-optimization based on QoS constraints. The traffic engineering techniques involved with p2mp TE LSP computation involve mainly with the case of p2mp path computation over multiple domains. There are three main issues involved with this feature: 1. load sharing among paths, 2. ability to modify the p2mp paths in different PCEDAs even when the PCEDA entities lie in different multiple domains, 3. p2mp path computation for corresponding clients in muliple domains must be able to support scalability, i.e. the number of clients entering/leaving the p2mp tree at a given time.

3 Analysis of QoS Based p2mp Path Computation Using PCEMP

This protocol driven inter-domain network environment architecture requires performance analysis techniques that can accurately model QoS driven protocols. In our model we adapt a technique that is a modification of a trajectory splitting simulation technique based on direct probability redistribution (DPR) [6,7]. We have developed techniques and demonstrate their utility by applying DPR to a network model that includes inter-domain autonomous systems, and a detailed PCEMP protocol. DPR was found to work best in settings where equalizing the number of samples in each of

the subsets (subsets of the state space of the system) for probability tree redistribution occurred. In our model, derived from the PCEMP State Machines design, probability is redistributed from the subsets of states with high probability to the subsets with small probability. One consequence of this redistribution of probability is a decrease in the accuracy of estimates corresponding to high probability subsets. We now see how QoS based TE LSP path computation can be modeled using this concept.

3.1 Simulation of QoS Based p2mp TE LSP Path Computation Using PCEMP

In this section we will explain the basic principles of DPR -based splitting, a more complete presentation can be found in [6], [7]. In DPR, we partition our state space S into m mutually exclusive, non-empty subsets $S_1, S_2, \cdots, S_m$. Each observation of the simulation is mapped into the appropriate subset according to function $\Gamma(X_i)$ which is given by,

$$\Gamma(X_i) = j,\ j = 1, 2, \cdots, m \tag{1}$$

where $X_1, X_2, \cdots, (X_i \in 1, 2, \cdots, m)$ are the observations of a discrete-time Markovian system. DPR modifies that transition probability matrix of the system such that each state has a new steady-state probability which can be written as,

$$\pi_i^* = \theta \pi_i \mu_{\Gamma(i)},\ \ i = 1, 2, \cdots, n \tag{2}$$

where n is the number of states, and π_i and π_i^* are the steady state probabilities before and after the DPR process for state i. θ is the normalization constant which is given by,

$$\theta = \frac{1}{\sum_{i=1}^{n} \pi_i \mu_{\Gamma(i)}} \tag{3}$$

The oversampling factor $\mu_{\Gamma(i)}$ is the main parameter that controls the modification of the steady-state probabilities. We assume that $\mu_1 = 1$ and $\mu_1 \leq \mu_2 \ldots \leq \mu_m$, which can be achieved for arbitrary ordering of the subsets by changing the ordering and the indexes of the oversampling factors (μ values).

3.2 Functional Parameters for p2mp TE LSP Computation in PCEMP Structures

The input data sequence is arranged into an ordered set called the Input Data Type (IDT) which is a subset of the input vector S and a function of the network transform to be computed T. A State Subset is a member of the cardinal product of S and T. It is shown to be isomorphic with the logical decoder outputs [3, 9]. The IDTs invoke the hardware for computing across the partitioned kernel in the PCE nodes.

Input: IDT T_j, State Subsets S_l and S_m, Integers l and m, Label L_b, Semi-Ring R. For p2mp support, there will be multiple state subsets, and we will pairwise consider all such states. In case where the total number of states is odd, one state will be paired with the identity state.

Output: Flow metric/measure p(A,B), which maps to the PCE descriptor ID. For p2mp cases, the PCE descriptor IDs are setwise collected to form the pp2mp ID.

3.3 QoS Based Path Computation Using PCEMP

Concept: Iterative applications of the PCEMP DS. Two or more IDT encoders separated by an interleaver (respectively CC and SPC). This structure suggests a decoding strategy based on the iterative passing of soft-computing information between two decoding algorithms. This is equivalent to dynamically partitioning the path computing engine core into sub-units that are linked together real-time based on the input data and the protocol handler. This is what makes PCEMP a protocol driving architecture, and is one of the key features of realizing a NP-hard path computation for p2mp TE LSPs.

Basic Computation: Configure PCEMP DS to compute soft-decisions in terms of measures. An assigned measure is given to each branch of the IDT. This algorithm makes the data intensive path computing much easier and reduces overhead and complexity and is incorporated in the computing core. It also guarantees disjoint path computation that enables fast end-to-end backup and protections. The configuration is totally dependent on the processed data and in a PCE server based bandwidth-on-demand scenario, can be triggered by the QoS service classes. The QoS classes are directly mapped onto the IDT, and thus can realize the p2mp based TE LSP path computation and re-optimization all the time based on the demanded bandwidth ensuring robustness and reliability of services. This follows directly from the PCEMP protocol architecture, details of which can be found at [3].

Section of the pseudo-code for the PCEMP FSM execution algorithm, Guha, Dipnarayan et al, Path Computation Element Metric Protocol, IETF Internet Draft, July 2005

```
for
{
any TE LSP passing through a PCE node P
{
Initialization..
Loop..

//QoS based p2mp TE LSP path computation support BEGIN //
do
{
repeat for all Si's;
assign boutS for each si in Si for P;
}
//QoS based p2mp TE LSP path computation support END //
..
//QoS based p2mp TE LSP path computation support BEGIN //
```

```
do
{
repeat for all Si's;
assign p for all c0's for all Si's;
take the weighted arithmetic mean of the probabilities
along with the branch labels;
assign p1 = probability weighted (mean(c0));
}
// QoS driven p2mp TE LSP path computation support END //
```

$\log p(S_i, x_j) = \log sum(s_i, x_{j-1}) + \log sum(z_j)$, $for \forall b \in B_i$ . .

$p(s_i, x_j) = \min \log(sum(s_i, x_{j-1}))$ .. Stop

3.4 Cramer Rao Bound Considerations for the QoS Based PCEMP Algorithm

With reference to [3], the probability density function of each data block of the PCE input c1, chosen without loss of generality, conditioned on the variable c0, the common symbols between the two encoders, CC and SPC and the other PCE input x1 is given by

$$p_{c_1/c_0,x_1}(c_1(n)/c_0, x_1(n)) = \frac{1}{(2\pi\sigma)^N} e^{-|c_1(n)-C_0 F x_1(n)|^2/(\sigma)^2} \tag{4}$$

where N is any arbitrary integer representing the cardinality of the data blocks, σ being the standard deviation of π_i, the steady state probability that the spanning tree is in state i, as in (2), and F is a unitary matrix whose rank varies as a function of i and N. C_0, F, x_1 are convoluted together. As the cardinality of the data blocks are uncorrelated, the joint pdf of a block $C_{1M} = [c_1(1),\ldots\ldots c_1(M)]$ is given by:

$$p_{C_1(M)/C_0,X_1(M)}(C_1(M)/C_0, X_1(M)) = \frac{1}{(2\pi\sigma)^{NM}} \exp(-\sum_{n=1}^{M} | c_1(n) - C_0 F x_1(n) |^2 / \sigma^2) \tag{5}$$

To simplify the Cramer Rao Bound derivation for determining the PCEMP IDT map and the optimal number of PCEDAs to be allotted to a PCE node, we now introduce a parameter vector θ, as

$$\theta = [c_0^T, x_1^T(1), ..., x_1^T(N), c_0^H, x_1^H(1), ..., x_1^H(N)] \tag{6}$$

and write the log-likelihood function as (6)

$$f(\theta) = \log p_{C_1(M)/C_0, X_1(M)}(C_1(M)/C_0, X_1(M)) = -\log((2\pi\sigma)^{MN}) - \sum_{n=1}^{M} \frac{| c_1(n) - C_0 F x_1(n) |^2}{\sigma^2} \tag{7}$$

This θ is related to the state probabilities as in (3).

We define L as the instantaneous data block partitioning that the CC and SPC enforces on the data block currently under processing, i.e. the cardinality of the data blocks are further partitioned into L for the purpose of parallel processing. It becomes easy to model equation (6) for the bound by introducing the 2(L+1)NM X 2(L+1) NM complex Fisher's Information Matrix, as in [8], thus

$$J(\theta) = E_{C_1(M)/C_0, X_1(M)} = \{(\frac{\partial f(\theta)}{\partial \theta^T})^H \frac{\partial f(\theta)}{\partial \theta^T}\} \tag{8}$$

where $\frac{\partial f(\theta)}{\partial \theta^T}$ is a 1 X 2(L+1)NM row vector. It has been proved in [8] that the CRB of θ can be found by considering the reduced (L+1)NM X (L+1)NM matrix thus:

$$J'(\theta) = E_{C_1(M)/C_0, X_1(M)}\{(\frac{\partial f(\eta)}{\partial \eta^T})^H \frac{\partial f(\eta)}{\partial \eta^T}\} \tag{9}$$

where $\eta = [C_0^T, X_1^T(1), \ldots X_1^T(M)]$. [8] goes on to derive the CRB for a fixed N, L and M. In our case, we assume that N and L are only fixed for a given observation time interval, that is, when the first route tree alignment corresponding to the first state happens. The model in [8] thus becomes finitely discontinuous at the end of each observation time interval. This doesn't prevent us from calculating the reducing CRB for the PCEDA allocation optimum. What we do is introduce a matrix W' and relax two restrictions which have been assumed in [8], 1. The size of the matrix can be variable. For the purpose of considering a restricted observation time interval and for applying [8]'s model on the piecewise continuous bound in time, we pad the matrix W' with ones so that there is no change in its' overall rank. This essentially means to mask the data block bits out by a series of 1's in the CC-SPC unit so that the overall logic function does not change. 2. The second relaxation is that the columns of W' need not form an orthonormal basis for the null space of C_0. In our masking bit scheme, it is possible to derive a secondary matrix W'' from W' that forms this basis, which is implemented into the processing hardware that is programmed with transforms, as in [9]. We thus have, from (1), and along the lines of [8], defining L' as derived from the bit masking process,

$$C = \frac{W'}{\sigma^2}\{W'^H \Pi I_{L'XL'} \otimes (F^* X_{1M}^* X_{1M}^T F^T)]\Gamma^H W'\}^{-1} W'^H = \frac{1}{\sigma^2}\Pi[I_{L'XL'} \otimes (F^* X_{1M}^* X_{1M}^T F^T)^{-1}]^{-1}\Gamma^H \tag{10}$$

It is easy to see that the maximum value of this function (9) is 1. We had started our analysis with c_1, with one of the triads of (c_0, c_1, x_1) arbitrarily without loss of generality. The elements of this triad are independent of each other, and thus, the optimal bound of the data-driven computation function is thrice the value of C that we have computed in (9), which effectively means, that if there is one-to-one map from the PCEMP IDT to the number of PCEDAs allocated to a PCE node, the maximum

number of PCEDAs optimally handled for best TE LSP path computation is 3. We shall confirm this result in this paper in our simulations section.

3.5 QoS Oriented Operation Considerations of p2mp in PCE

As we have said before, it is possible to obtain a protocol driven network architecture from a data driven protocol FSM. From the operation point of view, there are two equally likely possibilities for QoS oriented p2mp support in PCEs:

1. The PCE descriptor ID that is obtained from the FSM execution can be carried as a separate optional object in standard OSPF/RSVP-TE extensions, irrespective of whether a routing or a signaling based solution is deployed for TE LSP path computation in p2mp scenarios. Traditionally, if an explicit-route object is used, the PCE descriptor ID can be used in conjunction with it as a sub-object. It is easy to understand that a path change will essentially mean changing the contents of the explicit-route object and/or inserting/deleting a new one for the purpose of p2mp support. The pp2mp ID, which is added on once the PCE descriptor IDs are added with the explicit route object being processed by every next hop, will be thus spanned over the entire PCEDA. At the egress side, the total pp2mp ID is recognized and the LSP contents mapped onto the corresponding client paths.

2. The other option is to have a separate messaging system for PCEMP that only has the PCEMP header and the PCE descriptor ID as the payload. The PCE nodes can maintain a local counter for these IDs, which are generated randomly but become fixed for any set of adjacent path computation. The scope of this deployment is again implementation specific. It might act as an encapsulated packet within standard routing or signaling protocols, or may be run independently before control and management information is exchanged, or may be periodically run to maintain the "soft-state" like conditions. In either case, the p2mp TE LSP path computation is independent of the number of clients (or end points) that are attached to the PCE node, resulting in clear scalability enhancements. It is also evident that the make-before-break conditions in modifying p2mp TE LSPs can be easily done without much overhead and computation intensive operations.

Based on the event states of the protocol, the corresponding trajectory partitioning is mapped onto the PCEDAs, enabling a correct performance analysis model for PCEMP based inter-domain architectures. Including this correlation, the estimator for the variance in PCEMP IDT maps, from (1), (2), (3), (6) and (10) is

$$\begin{aligned} \mathrm{var}(\hat{\pi}_i) &= \mathrm{var}\Big(\sum_{k=1}^{N} \Lambda_i(X_k)/N\Big) \\ &= \frac{1}{N^2}\mathrm{var}\Big(\sum_{k=1}^{N} \Lambda_i(X_k)\Big) \\ &= \frac{1}{N^2}\left\{ N\sigma_i^2 + 2\sum_{j=1}^{N-1}(1-j/N)\rho_j\sigma_i^2 N \right\} \\ &= \frac{\sigma_i^2 R}{N} \end{aligned} \tag{11}$$

3.6 Simulation Results

We have defined a set of PCEMP messages and their types [3] recently at the IETF PCE WG, where we have used 8 different test case scenarios for the PCEMP driven inter-domain network environments where we performed queuing analysis and timing delay analysis. The simulations were carried out by developing external process modules for OPNET Modeler 8.0 Here are some of the results that we got by porting these external process modules to OPNET Modeler 8.0. The equations for modeling are based on (1), (2), (3), (4), (5), (10) and (11). We have developed a path computation and routing protocol software for this purpose based on our analysis and the definition of the protocol [10].

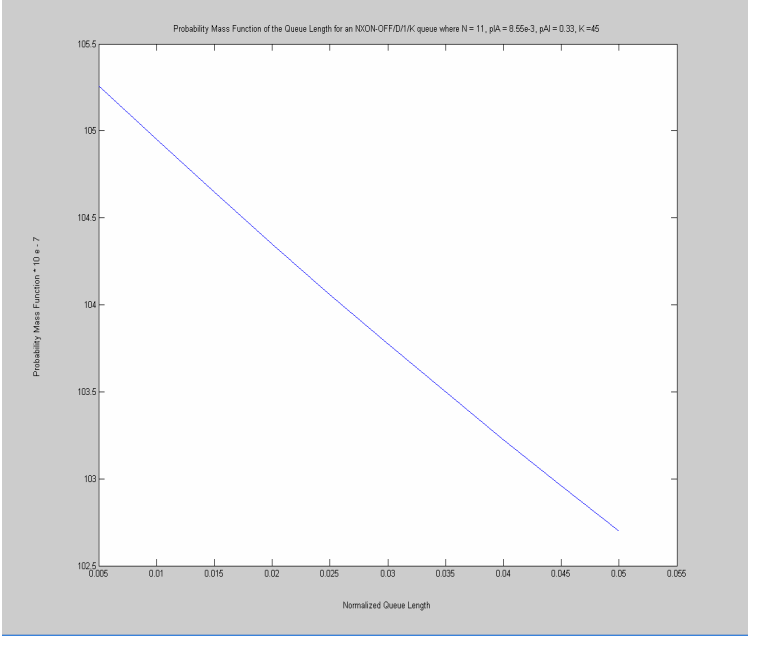


Fig. 1. PMF of the M/D/1/K model of the PCE architecture vs. normalized queue length

The first result is a classic where we get the results for the Probability Mass Function (PMF) of the NXON-OFF/D/1/K queue for N = 11, pIA = 8.55e-3, pAI = 0.33 and K = 45. The values were considerably in the region of 10e-7 for different normalized queue lengths, and though the trend was similar to [6] and [7], the results used in the PCEMP context seem to be better and provide a better upper bound to the optimal stable queue length. The piecewise continuous PMF over the observation interval is deduced from equations (4) and (5). Equations (6), (7), (8) and (9) help in computing this function and port it onto PCE software.

The second simulation result is carried out using equations (10). As discussed in section 3.4, this shows why the optimal number of PCEDAs is 3, based on the independence of the triad (c_0, c_1 and x_1).

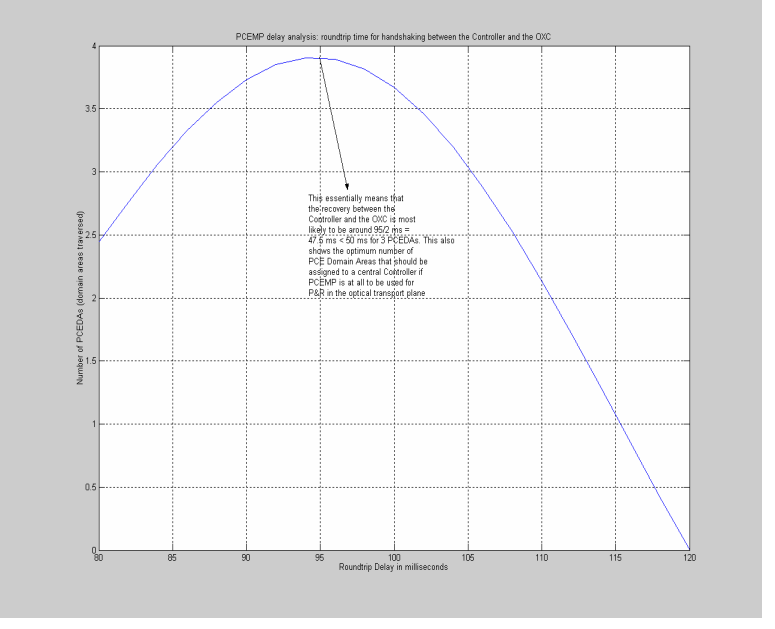


Fig. 2. Optimal number of PCEDAs based on the Cramer Rao bound from QoS based PCEMP

4 Conclusion

PCEMP helps the p2mp participating nodes to advertise their capabilities based on the set of constraints that it can support. Using PCEMP for p2mp TE LSP path computation and global re-optimization also serves the dual purpose of topology reconfiguration. This paper provides a general framework for p2mp support in PCE architectures, and is based on a scenario where the p2mp path computation is triggered by a QoS requirement. The authors are pursuing the standardization of this protocol in the PCE WG for an efficient inter-domain TE LSP path computation, and this is under current discussion within the scope of the newly constituted PCE WG at the IETF.

Acknowledgement

This work was supported by the in part by the Korea Science and Engineering Foundation (KOSEF) and the Institute of Information Technology Assessment (IITA) through the Ministry of Information and Communications (MIC), Republic of Korea.

References

1. Farrel, A., Vasseur, J.P., Ash, J.: Path Computation Element (PCE) Architecture, IETF Internet Draft,July2005, http://www.ietf.org/internet-drafts/draft-ietf-pce-architecture- 01. txt
2. Yasukawa, S. (Ed.): Signaling Requirements for Point to Multipoint Traffic Engineered MPLS LSPs, IETF Internet Draft, June 2005, http://www.ietf.org/internet-drafts/draft-ietf-mpls-p2mp-sig-requirement-03.txt

3. Choi, J.K., Guha, D.: Path Computation Element Metric Protocol (PCEMP), IETF Internet Draft, July 2005, http://www.ietf.org/internet-drafts/draft-choi-pce-metric-protocol-02.txt
4. Choi, J.K., Guha, D., Jo, S.K.: Considerations of point-to-multipoint route optimization using PCEMP, IETF Internet Draft, July 2005, http://www.ietf.org/internet-drafts/draft-choi-pce-p2mp-framework-01.txt
5. Choi, J.K., Guha, D., Jo, S.K., Cuong, D.H., Yang, O.S.: Fast end-to-end restoration mechanism with SRLG using centralized control, IETF Internet Draft, July 2005, http://www.ietf.org/internet-drafts/draft-choi-pce-e2e-centralized-restoration-srlg-03.txt
6. Akin, Y., Townsend, J.K.: Efficient Simulation of TCP/IP Networks characterized by non-rare events using DPR based splitting. IEEE Computer Society 2001, vol. 3, pp. 1734-1740
7. Nakayama, M.K., Shahabuddin, P.: Quick simulation methods for estimating the unreliability of regenerative models of large, highly reliable systems. ACM source probability in the Engineering and Informational Series archive, vol. 18, issue 3, July 2004, pp. 339-368
8. Barbarossa, S., Scaglione, A., Giannakis, G.: Performance analysis of a deterministic channel estimator for block transmission systems with null guard intervals. IEEE Transactions on Signal Processing, vol. 50, number 3, March 2002, pp. 684-695
9. Guha, D.: An interesting reconfigurable optical signal processor architecture. Proceedings of SPIE, Vol. 5246, pp. 656-659, August 2003
10. Guha, D., Choi, J.K., Jo, S.K., Cuong, D.H., Yang, O.S.: Deployment considerations of Layer 1 VPNs using PCEMP. IEEE INFOCOM 2005 Poster/Demo Session, Miami, FL, U.S.A., March 13-17, 2005, http://dawn.cs.umbc.edu/INFOCOM2005/guha-abs.pdf

Lightweight Real-Time Network Communication Protocol for Commodity Cluster Systems*

Hai Jin, Minghu Zhang, Pengliu Tan, Hanhua Chen, and Li Xu

Cluster and Grid Computing Lab.,
Huazhong University of Science and Technology, Wuhan, 430074, China
hjin@hust.edu.cn

Abstract. Lightweight real-time network communication is crucial for commodity cluster systems and embedded control systems. This paper introduces the design, implementation and evaluation of SS-RTUDP, a novel zero-copy data path based real-time communication protocol with efficient communication resources management. To avoid unpredictable overheads during SS-RTUDP packets transmission, all communication resources are pre-allocated. A feasible fragmentation mechanism is also proposed for transmitting SS-RTUDP packets larger than the network *MTU*. On the other hand, the additional real-time traffic smoother provides high priorities to SS-RTUDP packets and also smoothes peak packets arrival curve. The prototype of SS-RTUDP is implemented under Linux systems and performance evaluations over Fast/Gigabit Ethernet are provided. The measurement results prove that SS-RTUDP can provide not only much lower latency and higher communication bandwidth than traditional UDP protocol, but also good real-time network communication performance for commodity cluster systems.

1 Introduction

Network environment within cluster system has changed to Fast/Gigabit Ethernet with higher bandwidth and lower error-rate. Thus, the user expects mostly in cluster systems is the network end-to-end communication performance. The performance of the communication sub-system in a cluster system is determined by the transmission rate of the network hardware, the processing ability of I/O buses, and also the software overheads of network protocol. In recent years, while the network hardware and I/O buses provide gigabit bandwidth, the traditional software overheads of network communication protocol (such as TCP/IP stack) have become the bottleneck, especially in cluster system with large amount of message passing [1].

To improve the communication performance in cluster system, several approaches have been considered, such as using multiple parallel networks, implementing lightweight protocols, avoiding data buffering and copying, avoiding system call invoking, overlapping communication and computation, using fast interrupt paths or avoiding interrupt processing, and introducing Jumbo Frames [2].

* This paper is supported by National 863 Hi-Tech R&D Project under grant No.2002AA 1Z2102.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 1075 – 1084, 2005.
© IFIP International Federation for Information Processing 2005

Two main approaches adopted to reduce the software overheads of a communication protocol are the improvement of the TCP/IP layers and the substitution of the TCP/IP layers by alternative ones [3]. The former focuses mainly on *zero-copy* architectures [4][5], which are capable of moving data between application domains and network interfaces without any CPU and memory bus intensive copy operations. Two alternatives can be considered to the latter approach: communication protocol with efficient OS support [6][7], and the user-level network communication [8]~[13]. But most of these facilities require special NIC hardware.

In this paper, we propose SS-RTUDP (***S**imple and **S**tatic-resources-allocation **R**eal-**T**ime **U**ser **D**atagram **P**rotocol*), a lightweight real-time network communication protocol over commodity Fast/Gigabit Ethernets. High data throughout, low latency and real-time communication are the major requirements for SS-RTUDP and the zero-copy data path and a fragmentation mechanism are adopted in SS-RTUDP. To satisfy real-time communication requirements, an adaptive real-time traffic smoother is employed and all communication resources are pre-allocated.

The organization of this paper is as follows. Section 2 describes the processing of original UDP protocol. Section 3 introduces the design and implementation of SS-RTUDP in detail. The performance measurements of SS-RTUDP prototype system are shown in Section 4. Section 5 states the conclusions and directions of future work.

2 Overhead Analysis of UDP Protocol

The overhead of UDP protocol consists of per-packet and per-byte costs [14]. The per-packet costs include the OS overheads and the protocol processing overheads. The per-byte costs include data-copying overheads and checksum calculation overheads. Fig. 1 shows the data and control flow of UDP/IP processing on Linux system. These costs are analyzed using the Intel PRO/1000 Gigabit Ethernet card with Intel Celeron 2.0GHz PC (RedHat9.0, Linux 2.4.20 kernel). Table 1 shows the total overhead during sending and receiving a UDP packet over Gigabit Ethernet. The most dominant overheads are for UDP/IP layer processing, including memory data copies, checksum calculations, resources allocating, etc.

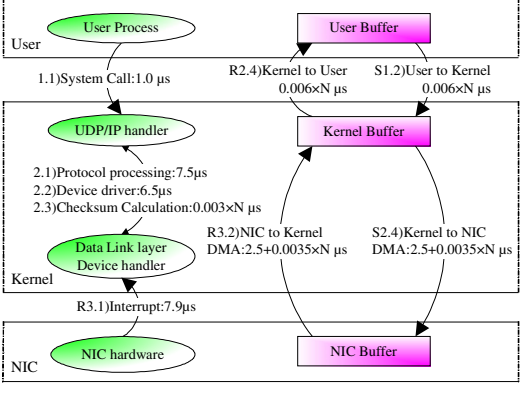


Fig. 1. Data and control flow and their costs using UDP protocol

Table 1. Total cost evaluation of UDP processing

Overheads list	Gigabit Ethernet	
	Costs(μs)	***(%)***
System call and socket layer	***3.2***	***4.7%***
UDP/IP layer(sender)	***14.8***	***21.7%***
UDP/IP layer(receiver)	***23.5***	***34.5%***
Device driver layer	***6.8***	***10.0%***
NIC interrupt processing	***6.9***	***10.1%***
DMA and media transmit	***5.5***	***8.1%***
Others	***7.4***	***10.9%***
Total	***68.1***	***100%***

3 SS-RTUDP Approach

The SS-RTUDP design issues can be summarized as follows:

- Use commodity hardware and without modifying hardware firmware.
- Minimal modification to the Linux kernel and good portability to other systems, such as real-time systems and embedded control systems.
- Coexisting with standard protocol such as TCP/IP stack.
- Lightweight communication guarantee (zero-copy data path).
- Provide soft real-time communication.

3.1 Network Communication Architecture with SS-RTUDP

The network communication architecture with SS-RTUDP is shown in Fig. 2. Three main parts are involved in SS-RTUDP protocol: user-level application library, SS-RTUDP/IP layer and real-time traffic smoother.

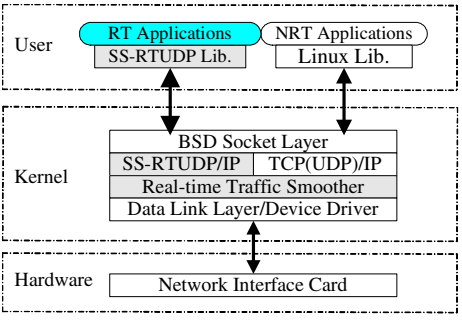


Fig. 2. Network communication architecture with SS-RTUDP protocol

3.1.1 Application Programming Interfaces (APIs)

In current SS-RTUDP communication library, five APIs similar to traditional communication APIs are provided: *rt_socket(), rt_mmap(), rt_sendto(), rt_recvfrom() and rt_socket_close()*. Table 2 shows the main functions of them.

Table 2. User-level APIs provided by SS-RTUDP

API	Function description
rt_socket()	*1) Apply for a new real-time socket* *2) Allocate appropriate kernel buffer for the socket* *3) Initialize a socket buffer pool for the socket* *4) Return the physical address of the kernel buffer allocated*
rt_mmap()	*Remap the user send-buffer space to real-time socket's kernel buffer*
rt_sendto()	*Transmit a SS-RTUDP packet*
rt_recvfrom()	*Receive a SS-RTUDP packet*
rt_socket_close()	*Close a real-time socket*

3.1.2 Zero-Copy Sending Data Path in SS-RTUDP

In SS-RTUDP protocol, the network *MTU* is 1500 bytes, the header length of IP protocol (H_{ip}) 20 bytes, the header length of SS-RTUDP protocol ($H_{SS\text{-}RTUDP}$) 8 bytes, and the Ethernet hardware header (H_{hh}) 16 bytes. A SS-RTUDP packet larger than *MTU* must be split into several IP fragments before transmission. The headers and the data of an IP fragment must be assembled within continuous physical address space due to *DMA (Direct Memory Access)* requirements. Fig. 3 describes the fragmentation mechanism, where L_{data} stands for data size of a SS-RTUDP packet to be sent while L_{left} the data size left to be sent, and pointer p points to head of the socket buffer.

1. Lock user send-buffer and set $L_{left}=L_{data}$.
2. Allocate a socket buffer from the socket's *rtskb_pool*.
3. Assemble the IP, SS-RTUDP and Ethernet headers from p.
4. If $L_{left} \le P_{max}$, transmit the single packet from p with $H_{ip}+H_{SS\text{-}RTUDP}+H_{hh}+L_{left}$ size using *DMA* to the network and jump to step9; Else, transmit the first fragment with $H_{ip}+H_{SS\text{-}RTUDP}+H_{hh}+P_{max}$ size from p to network using *DMA* and set $L_{left}=L_{left}-P_{max}$.
5. Set $p=p+1480$, and allocate a socket buffer from the socket's *rtskb_pool*.
6. Assemble the IP and Ethernet headers from p.
7. If $L_{left} \le 1480$, transmit the last fragment with $H_{ip}+H_{hh}+L_{left}$ size from p to the network using *DMA*; Otherwise, transmit the fragment with $H_{ip}+H_{SS\text{-}RTUDP}+H_{hh}+P_{max}$ size from p to the network using *DMA*.
8. Set $L_{left}=L_{left}-1480$ and jump to step 5.
9. Unlock user send-buffer.

Fig. 3. A fragmentation mechanism for SS-RTUDP protocol

Based on the fragmentation mechanism in Fig. 3, we design a zero-copy data sending path in SS-RTUDP shown in Fig. 4. The user send-buffer is remapped to the kernel buffer first using the *rt_mmap()* system call and all the IP fragments are assembled in continuous physical space without additional kernel buffer for the headers of each IP fragment.

3.2 Real-Time Communication Considerations for SS-RTUDP Protocol

Two aspects are considered to provide real-time performance for SS-RTUDP. One is to avoid dynamic communication resources allocation. The other is to add real-time traffic smoother to provide higher priority to SS-RTUDP packets than other packets.

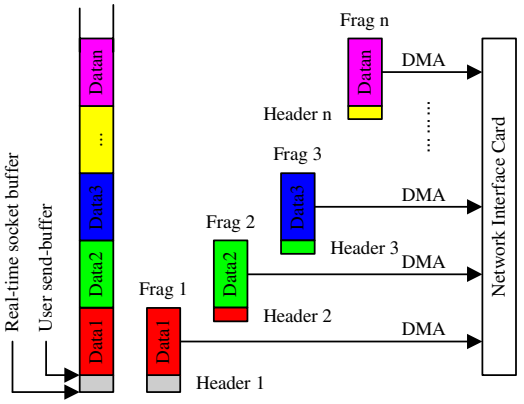


Fig. 4. Zero-copy data sending path in SS-RTUDP

3.2.1 Pre-allocation of Network Resources

To satisfy real-time communication performance of SS-RTUDP, the unpredictable overheads in the communication path must be avoided. The main unpredictable operation is dynamic kernel buffers allocation to buffer the data being copied from the user space. During the initialization of SS-RTUDP protocol, a global buffer pool is initialized, where four kinds of kernel buffer blocks (1, 4, 8 and 16 continuous physical memory pages) are pre-allocated and pinned down. The main functions of *rt_socket()* system call are to apply one appropriate kernel buffer block and initiate a socket buffer pool (*rtskb_pool*). Through remapping its send-buffer to the socket kernel buffer using the *rt_mmap()* system call, the user application can directly use the socket kernel buffer. The socket buffer needed during packet sending is not dynamically allocated but directly achieved from the *rtskb_pool*, and the buffer space of each socket buffer is set to appropriate position of the socket kernel buffer block.

3.2.2 Real-Time Traffic Smoother

The main functions of the real-time traffic smoother are to control the *non-real-time* (*NRT*) packet arrival rate to appropriate input limit, without affecting the *real-time* (*RT*) packet arrival rate. The real-time traffic smoother provides statistical real-time communication performance that the probability of packet lost is less than a certain loss tolerance, Z [15]:

$$Pr_{(packet\ loss\ rate)} \leq Z \tag{1}$$

The traffic smoother is leaky bucket-based [16], where *credit bucket depth* (*CBD*), the capacity of the credit bucket, and a *refresh period* (*RP*) are defined. Every *RP*, up to *CBD* credits, are replenished to the bucket. In our implementation, the unit of credit is packet. We set the input limit of *packet arrival rate* (*PAR*) as the value of *CBD/RP*, which determines the average throughput available:

$$PAR = CBD/RP \tag{2}$$

By fixing the values of *CBD* and varying the values of *RP* ($RP_{min} \le RP \le P_{max}$), it is possible to control the burst nature of packets flows generated. We set RP_0 ($RP_{min} < RP_0 < RP_{max}$) as the initialization value (also the average value) of *RP*.

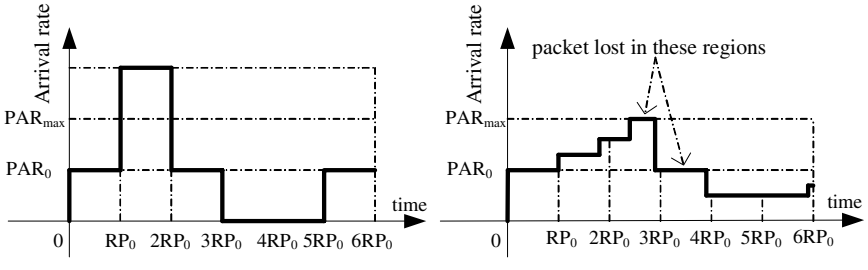
$$PAR_0 = CBD/RP_0 \tag{3}$$

Fig. 5 shows the smoothing and refreshing procedures provided by the real-time traffic smoother. When a *NRT* packet arrives from the IP layer, the traffic smoother sends it to the NIC and removes one credit if there is at least one credit in the bucket. Otherwise, the *NRT* packet is buffered. A *RT* packet is not affected by the traffic smoothing, but it consumes a credit also.

Smoothing	**Refreshing**
1 if(SS-RTUDP){	RP = RP_0;
2 send-to-NIC();	if (packet lost in the latest RP region)
3 CNS = CNS-1;}	RP = min(RP_{max}, 2×RP);
4 else {	else
5 if(CNS≥0){	RP = max(RP_{min}, RP-ΔRP);
6 send-to-NIC();	if (CurrentTime == NextRefreshTime)
7 CNS = CNS -1;}	{
8 else	CNS = min(CBD, CNS+CBD);
9 send-back-to-queue();	NextRefreshTime = CurrentTime+RP;
10 }	}
RP_0 : The initialization value of RP RP_{max} : the maximal value of RP RP_{min} : the minimal value of RP CNS : Current Number of Credits	

Fig. 5. Smoothing and refreshing procedures

Fig. 6 shows the effect of traffic smoothing. Fig. 6 (a) shows the symmetric burst input arrival rate during [*0, 6RP₀*] from the IP layer to the NIC and Fig. 6 (b) shows the smoothed output arrival rate from the NIC to the network.



(a) Input arrival rate (b) Output arrival rate

Fig. 6. Traffic smoothing results

4 Performance Analysis

All the evaluations of SS-RTUDP are made using a cluster with 24 nodes. Table 3 shows the configuration of the evaluation environment.

Table 3. Evaluation environment configuration

Hardware		***Intel Celeron 2.0GHz CPU; 256MB DDR Memory; 33MHz/32bit PCI bus***
NIC	***Fast Ethernet***	***3COM 3C905B 10/100M***
	Gigabit Ethernet	***Intel PRO/1000MT Gigabit Ethernet card***
		3COM 3c905B Fast Ethernet card
Ethernet switches		***D-link DES-1024$^+$Fast Ethernet switch***
Operating systems		***RedHat 9.0, Linux kernel 2.4.20***
Traffic smoother parameters		***CBD=40, RP_0=10ms, RP_{max}=100ms, ΔRP=2ms, RP_{min}=5ms***

The basic application level communication bandwidth and latency over Fast/Gigabit Ethernet and the real-time communication performance over Fast Ethernet, are measured and compared with traditional UDP/IP in this section. We use the modified NetPIPE [17] benchmark to evaluate network latency and bandwidth. The real-time performance is measured by packet loss rate.

4.1 Latency and Bandwidth

In this experiment, two nodes are connected directly by the cross-over cable. The latency and bandwidth results over Fast Ethernet are shown in Fig. 7.

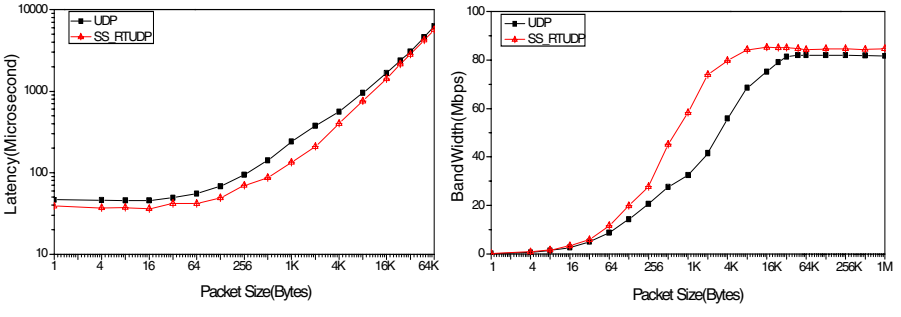


Fig. 7. Latency and bandwidth over Fast Ethernet

In Fig. 7, SS-RTUDP over Fast Ethernet can achieve 37μs latency in a 4 bytes message, which is a little smaller than 45.8μs of UDP. Especially, the latency decreases over 36% over SS-RTUDP ranging between 256 to 4096 bytes. Fig. 7 also shows that SS-RTUDP can achieve a maximal bandwidth of 89.6 Mbps which is a little higher than 81.9 Mbps of UDP. These results also show that SS-RTUDP can achieve half of the maximal bandwidth below 512 bytes packet size, while this value is over 2000 bytes in UDP. Obviously, the performance improved by SS-RTUDP protocol is limited over Fast Ethernet due to low media transmitting rate.

Fig. 8 shows the latency and available bandwidth achieved over Gigabit Ethernet. The results show that SS-RTUDP can exploit much more Gigabit Ethernet capacity than that of traditional UDP protocol. Over Gigabit Ethernet, SS-RTUDP can achieve 39μs latency in a 4 bytes message, which is much lower than 65.8μs of UDP (40.1% improvement), and a maximal bandwidth of 545.2 Mbps compared with 396.6 Mbps of UDP (37.5% improvement).

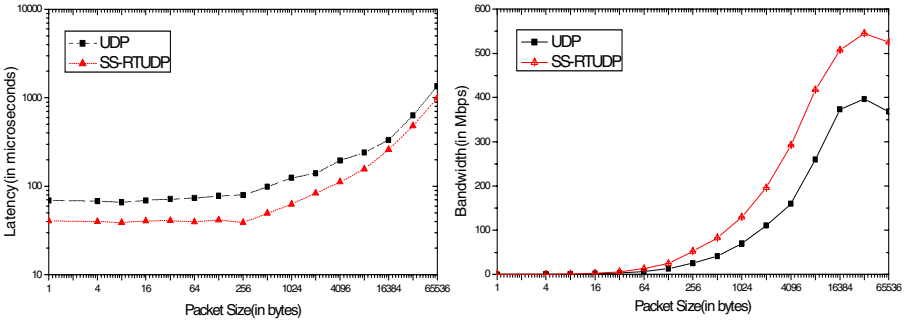


Fig. 8. Latency and bandwidth over Gigabit Ethernet

4.2 Real-Time Performance Evaluation

For two nodes connected directly, an additional long-time and high arriving rate TCP-stream (occupies over 80% of physical available bandwidth) is added. The experimental results in Fig. 9 show that even in very congested environment, SS-RTUDP can maintain normal and steady performance.

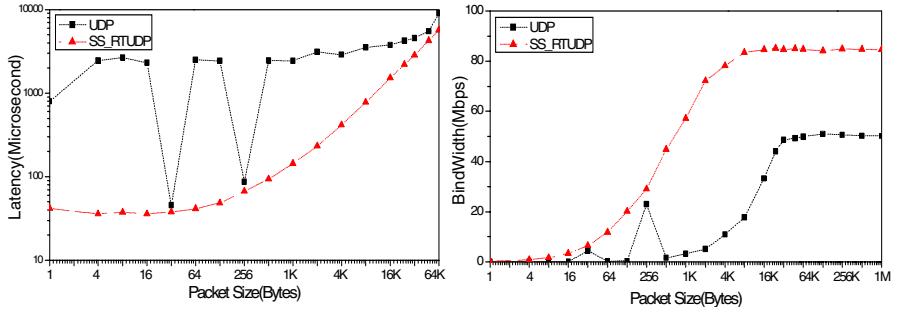


Fig. 9. Latency and bandwidth measurements under congested environments

When 4 cluster nodes are connected through the Ethernet switch shown in Fig.10, the total packet arrival rate from node 1~3 to node 4 through the switch is 105.51 Mbps. Based on the results of Fig. 7, we set 300μs as the transferring deadline to a RT packet and 1500μs to a NRT packet. Table 4 shows the experimental results.

Table 4. Packet loss rate measurement

Condition	Type	Sent	Lost	Loss rate
Without Traffic Smoother	RT	360000	56881	15.8%
	NRT	2250000	380250	16.9%
With Traffic Smoother	RT	360000	265	0.074%
	NRT	2250000	507329	22.6%

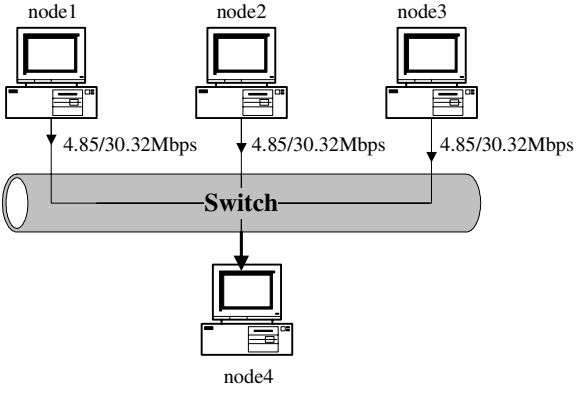


Fig. 10. Multiple nodes environment for real-time performance tests

From Table 4 we find that the *RT* packet loss rate reduces from 15.8% (without real-time traffic smoother) to 0.074% (with real-time traffic smoother), while the *NRT* packet loss rate increases from 16.9% to 22.6%. The results prove that the real-time traffic smoother provides good performance guarantee for real-time communications.

5 Conclusions and Future Works

In this paper, we propose a lightweight real-time network communication protocol for commodity cluster systems, called SS-RTUDP. Through eliminating data copies in communication data path, simplifying the data checksum calculations, adopting fragmentation mechanism for large real-time packets, SS-RTUDP provides both lower latency and higher bandwidth than original UDP protocol over Fast/Gigabit Ethernet, especially over Gigabit Ethernet. Pre-allocated network resources and additional real-time traffic smoother also provides good real-time network communication performance for commodity cluster systems.

Many works need to be improved to current implementation of SS-RTUDP, such as lightweight interrupt processing and true zero-copy data receiving path. In the next step, we will implement the SS-RTUDP protocol under real-time micro kernel to enhance the real-time performance over Gigabit Ethernets. Another goal is to plant SS-RTUDP protocol to embedded control systems.

References

[1] S. Di and W. Zheng, "Reduced Communication Protocol for Clusters", *Proceedings of Advances in Parallel and Distributed Computing*. Shanghai, China, 1997, pp.314-319.

[2] R. Buyya, *High Performance Cluster Computing Architectures and Systems*, Volume 1, 1st Edition, USA, Prentice-Hall, Inc., 1999, pp.182-184.

[3] F. Diaz, J. Ortega, A. Canas, F. J. Fernandez, M. Anguita and A. Prieto, "The lightweight protocol CLIC on Gigabit Ethernet", *Proceedings of International Parallel and Distributed Processing Symposium(IPDPS'2003)*, April 2003.

[4] K. Christian, M. Michel, and R. Felix, "Speculative Defragmentation – A Technique to Improve the Communication Software Efficiency for Gigabit Ethernet", *Proceedings of the 9th International Symposium on High-Performance Distributed Computing*, Pittsburgh, PA, USA, 2000, pp.131-138.

[5] K. A. Skevik, P. Thomas, G. Plagemann, and V. Goebel, and P. Halvorsen, "Evaluation of a Zero-Copy Protocol Implementation", *Proceedings of the 27th EUROMICRO Conference,* Warsaw, Poland, 2001, pp.324-330.

[6] G. Chiola and G. Ciaccio, "Efficient parallel processing on low-cost clusters with GAMMA active ports", *Parallel Computing*, Vol.26, 2000, pp.333-354.

[7] F. Díaz, J. Ortega, A. Cañas, F. J. Fernández, and A. Prieto, "The Lightweight Protocol CLIC: Performance of an MPI implementation on CLIC", *IEEE International Conference on Cluster Computing (CLUSTER'2001)*, October, 2001, pp.391-398.

[8] K. Ghouas, K. Omang and H. Bugge, "VIA over SCI - Consequences of a Zero Copy Implementation, and Comparison with VIA over Myrinet", *Proceedings of the 15th International Parallel and Distributed Processing Symposium (IPDPS'2001)*, San Francisco, California, USA, 2001, pp.1632-1639.

[9] Y. Chen, Z. Q. Jiao, J. Xie, Z. H. Du, and S. L. Li, "Design and Implementation of a High Performance VIA Based on Myrinet", *Journal of Software*, Vol.14, No.2, 2003, pp.285-292.

[10] S. Pakin, V. Karacheti, and A. Chien, "Fast Messages (FM): Efficient, Portable Communication for Workstation Clusters and Massively-Parallel Processors", *IEEE Parallel and Distributed Technlogy*, Vol.5, No.2, April/June, 1997.

[11] L. Prylli and B. Tourancheau, "BIP: a new protocol designed for high performance networking on Myrinet", *Proceedings of Workshop PC-NOW at IPPS/SPDP98,* April, 1998, pp.472-485.

[12] T. Eicken, A. Basu, V. Buch, and W. Vogels, "U-Net: a user-level network interface for parallel and distributed computing", *Proceedings of the 15th ACM Symposium on Operating Systems Principles (SOSP'95)*, December, 1995.

[13] R. A. F. Bhoedjang, T. Rühl, and H. E. Bal, "User-level Network Interface Protocols", *IEEE Computer*, November, 1998, pp.53-60.

[14] D. Clark, V. Jacobson, J. Romkey, and H. Salwen, "An Analysis of TCP Processing Overhead". *IEEE Communications Magazine*, Vol.27, No.6, June 1989, pp.23-29.

[15] C. Chou and K. G. Shin, "Statistical real_time channels on multiaccess networks", *IEEE Transaction on Parallel and Distributed Systems*, Vol.8, Aug. 1997, pp.769-780.

[16] R. L. Cruz, "A Calculus for network delay, Part I: Network Elements in Isolation", *IEEE Transaction on Information Theory*, Vol.37, No.1, Jan. 1991, pp.114-131.

[17] NetPIPE: A Network Protocol Independent Performance Evaluator, http://www.scl.ameslab.gov/netpipe/.

Towards a Secure and Reliable System

Michele Portolan and Régis Leveugle

TIMA Laboratory 46, Avenue Félix Viallet - 38031 Grenoble Cedex – France
Michele.Portolan@imag.fr, Regis.Leveugle@imag.fr

Abstract. In this article we describe a system based on a 32-bit processor, Leon, complete with security features offered by a specific cryptographic AES IP. Hardening is done not only on the principal hardware components but on the operating system as well, with attention for possible interaction between the different levels. The cryptographic IP is protected too to offer good resistance against, for example, fault-based attacks.

1 Introduction

Nowadays the application range of computing systems is steadily increasing, going well beyond classical applications, like for instance desktops or mainframes. This usually comes together with the search for extreme miniaturisation and low power, meaning that the resulting electronic circuits can be seriously affected by transient faults, once relegated to especially harsh environments (space, nuclear facilities, etc…). It is therefore important to provide hardening as soon as possible in the development process to assure good performances and fault coverage with contained costs. The usual hardening techniques, based on post-synthesis netlist modifications, can be really demanding in terms of overhead and, especially, development and verification time. One solution is to harden the design already at conception time, acting on the behavioural high-level description (usually in VHDL). These interventions can be a little more complex than their gate-level homologues, but have the great advantage of being directly verifiable in behavioural descriptions, for example through simulation, which is much lighter than post-synthesis validations. The result is the possibility of validating at an early stage the robustness of the protected system, instead of waiting after synthesis. Moreover it is possible to exploit the configurability features of these high-level languages to obtain solutions directly applicable to a large range of possible configurations, while gate-level hardening must be individually done for each netlist. The contribution of this paper is to illustrate ona practical case the feasibility and efficiency of this approach.

In this article we firstly introduce the concept of transient faults in section 2, pointing out the most common causes and fault models. Then we introduce a secure embedded system, detailed in section 3, whose hardening is then discussed in section 4. Section 5 evaluates the robustness level obtained, while lastly section 6 points out future developments.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 1085 – 1098, 2005.
© IFIP International Federation for Information Processing 2005

2 Natural and Provoked Faults

Transient faults are perturbations of the normal operation of the circuit that if not properly handled could cause erroneous results, with serious consequences. There can be many causes, like for instance the impact of a charged alpha particle causing an ionization of the channel, or the crosstalk between two adjacent wires. Electromagnetic fields could also alter the thresholds of transistors, while strong noises on the input signals may cause incorrect information to be sampled. Miniaturisation means higher density of the circuit, making it more sensitive to particle impacts and crosstalks, while low power means low working voltages and currents, making electromagnetic perturbation and noise effects more important too. Handheld devices for instance are especially exposed: on the one hand battery life (low power) and size (miniaturisation) can become key factors for the success of a product, while on the other hand they are even more sensitive to interferences while working near to other components. Anyone, for instance, has experienced the problems a mobile phone ringing can provoke to a stereo or a television.

Regardless to their cause these faults can be usually modelled as unwanted transitions either in the combinatorial logic (the result of the operation is not correct) or in the sequential logic (the value stored in a flip-flop is modified): if a single signal is modified they are called Single Event Transients (SETs) and Single Event Upsets (SEUs) respectively. They alone can model a great part of natural faults, while combining them together it is always possible to investigate more complex phenomena that involve errors with higher multiplicity.

As sensitive information is increasingly stored in portable devices (addresses, telephone numbers, credit card identifiers, etc …) there is also a rising need for security inside systems, often equipped with cryptographic units. In later years the emergence of fault-based attacks on cryptographic systems has cleared a point: reliability is important for secure systems too. The principle of a fault-based attack is to deliberately cause a fault in the system in order to obtain sensitive information that should have been kept secret. A famous example is the attack on the RSA algorithm implemented with the Chinese Reminder Theorem exposed by Lenstra in [1]: with one or two well-timed faults the circuits directly yields the secret key.

There is a great range and variety of fault-related attacks on secure systems: a complete panorama is out of the scope of this article, so we will just give a brief summary. In general terms attacks can be divided in three categories, based on their approach to the system they want to violate: invasive attacks, non-invasive attacks andsemi-invasive attacks.

Invasive attacks are based on physical modifications of the system, like inserting probes, cutting lines or short-circuiting some signals.

Non-invasive attacks do not physically modify the system but try to infer sensitive information either by observing its normal behaviour (side-channel attacks) or by perturbing it with the insertion of voluntary faults (fault-based attacks). Side-channel attacks exploit information that “leaks out” of the system by comparing runs on different loads of data. The most important are Timing Attacks (TA, time difference in calculations with different data), Simple and Differential Power Attacks (SPA and DPA, power consumption analysis) and Electromagnetic Attacks (EMA, analysis of the electromagnetic emissions by the system). On the other hand fault-injection

attacks presume the active intervention of the hacker who deliberately inserts a fault at a given instant to disrupt the normal execution of the cryptographic algorithm. If the injection instant is well calibrated the result can be quite effective, as previously illustrated for RSA.

Faults can be injected in a variety of ways, usually either by inserting glitches on the power source or the clock or by flashing the device (either the entire chip with a photolamp flash or more selectively with a laser beam).

Lastly semi-invasive attacks are between the two: sometimes in order to perform attacks it is necessary to prepare the circuit, usually decapsulating it from the protective resin to expose the target area to the light. As the chip itself is not modified such attacks are conceptually similar to the non-invasive ones.

The attention is put more on selective attacks provoking limited errors because large-scale ones can be easily blocked by sensors put on the chip that reset the system when, for example, a flash of light is detected. These devices on the other hand become useless against, for instance, a laser focalised on specific point of the system. This category of faults can once more be modelled in terms of SETs and SEUs, so the protection from natural and provoked faults can be merged. Note that when the fault model requires more than one error occurring simultaneously it may be possible in some cases to account for as many concurrent SETs/SEUs as needed.

In the following we will therefore focus on protection against SETs and SEUs, that may be either natural or provoked, without distinguishing further the two cases. Some multiple error configurations are implicitly also covered by the proposed implementation, but a complete protection against multiple-bit errors will not be discussed here.

3 Proposed Prototype System

3.1 System Overview

The system we propose, illustrated in Fig. 1, is based on a reconfigurable RTL processor core, Leon 2 [2], equipped with a cryptographic coprocessor implementing AES to provide security. The system runs eCos, an embedded Operating System (OS) by Red Hat [3], thus obtaining a complete platform well representative of Systems on Chip (SoCs). A real application could be obtained directly implementing it, while vice versa any typical single-processor system could easily be retraced to our example, making it possible to apply the proposed strategy on it.

3.2 Leon Processor

Leon is a SPARC V-8 compliant 32-bit RISC processor that is freely available on the net ([2]) as a VDHL synthesisable model covered by the GNU GPL. Such a format allows the processor to be freely tailored-out to suit the needs of the particular application, as well as being target of reconfiguration operations like those proposed in [4]. A block diagram of the processor is reported in Fig. 2. The centre of the processor is a 5-stage Integer Unit (IU) implemented with cache-level Harvard architecture (i.e. separate caches for data and instruction, but common central memory). Around that block many entities can be instantiated to adapt the processor

to the designer needs, notably an optional multiplier, a Debug Support Unit (DSU) to allow on-line debug through a serial link, a Memory Management Unit (MMU), an SDRAM controller, a Floating Point Unit (FPU), support for a Coprocessor (CP), PCI and Ethernet interfaces.

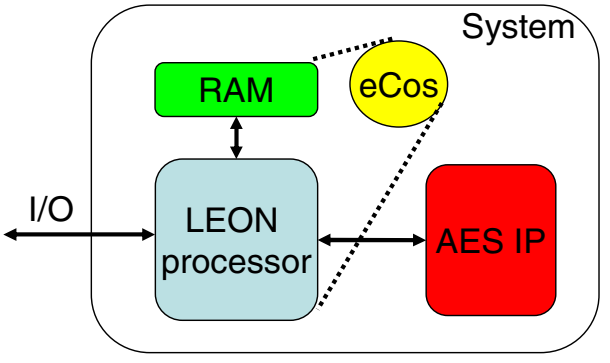


Fig. 1. Scheme of the proposed prototype system: the three main components (Leon processor, AES cryptographic IP and Operating System) and their interconnections are shown

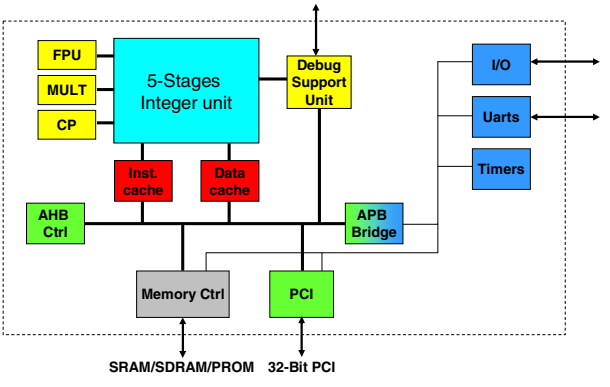


Fig. 2. Leon 2 Processor Block Diagram: note in particular in the upper left-hand corner the Integer Unit and the two caches (instruction and data)

From our point of view Leon offers many interesting points:

- it is well-known, widely tested and centrally maintained, offering a good applicability range;
- the GNU GPL allows us (almost) complete freedom of intervention on the source code;
- configurability is a key feature of the project;
- the core is build around two AMBA-compliant buses [5] (an AHB and an APB), making it easy to add new components.

We exploited these characteristics to enhance the processor and add new features, such as hardening and cryptographic capabilities. Leon has been developed with space applications in mind and so has already undergone hardening stages, resulting in the Leon-FT [6] that is also the base for a commercial product by Atmel, named AT697E [7]. Anyway its sources are not public, making impossible to directly use it: we decided to develop our own protections using the Leon-FT as an inspiration and comparison model.

3.3 AES Cryptographic IP

Portable (and therefore ubiquitous) systems are being used nowadays also as terminal for secure transactions, like for example biometric identification devices or bank payments. An example can be the Thumbpod presented in [8], where a Leon is coupled to a fingerprint scanner and an AES core to obtain a portable biometric identification unit. High connectivity capability (usually wireless) is becoming a standard requirement and, as recent years have shown, this too cannot come without proper security, especially when it comes to communications between peer devices. Unfortunately the two most classical wireless communication standards, Bluetooth and IEEE 802.11, have both great problems in that field, as can be seen for example in [9]: a good way to counter weaknesses in the standard is to boost up security in devices.

Security features have been added to our system by providing an optional cryptographic IP, realising AES encryption/decryption. Plugged on the APB bus of the Leon it permits fast secure operations, like for instance digital signatures or transactions: with that equipment our system is up to the challenge, remaining open to new cores implementing different algorithms. The core can be configured to support keys of 128, 192 or 256 bits (or all three) and the S-Boxes can be either hardwired in ROM or implemented in RAM (for Virtex II only). The IP has been synthetised with Leonardo for various configurations, the most representative of which are reported in Table 1 for the ROM-based implementation and in Table 2 for the RAM-based one.

Both implementations offer good results, in particular the ROM-based one has really good timing performances. On the other hand the RAM-based version can

Table 1. Synthesis Results for AES IP on Virtex-II, S-Boxes implemented in ROM

Key Size (bits)	Clock Cycles (Coding)	Clock Cycles (Decoding)	Max Clock (Mhz)	Hardware complexity	
				FG	RAMs
128	70	130	44.29	8160	0
192	82	122	45.85	9460	0
256	104	142	42.86	9836	0
All	***	***	45.23	11692	0

***: depends on key size, equivalent to precedent results

Table 2. Synthesis Results for AES IP on Virtex-II, S-Boxes implemented in RAM

Key Size (bits)	Clock Cycles (Coding)	Clock Cycles (Decoding)	Max Clock (Mhz)	Hardware complexity	
				FG	RAMs
128	580	642	56.66	4454	16
192	584	624	47.98	5752	16
256	610	654	56.21	6124	16
All	***	***	47.28	7610	16

***: depends on key size, equivalent to precedent results

reduce the occupied area by almost half and the maximal frequency is some 10% higher, but it causes a great timing loss in terms of cycles per operation (a factor varying from 6 to 8 depending on the implementation). Probably it pays heavily the access protocol used by the internal RAM the S-box are implemented in, called up for each calculation. Better results could be obtained using more performing or specially-tailored RAMs on ASIC targets. For example burst access could come useful if the data in the S-Box were cleverly placed. This version can nevertheless be interesting for applications where AES timing is not critical, like for instance an identification device (needing only a few calculations). More heavy time-critical applications, requiring for example a coder for data streaming, would prefer the ROM-style.

3.4 eCos Operating System

eCos is a multithread operating system having configurability and portability as its key features. On the one hand these features enabled us to easily modify it following our hardening strategy, as explained in section 4.3. On the other hand its good portability offered us a fully working Leon version, with the possibility of exploiting the other ports to instantly transfer it on another system if needed.

4 Robustness

4.1 Hardening Strategy

The system implements a protection scheme that has in the Operating System its fulcrum, but without forgetting the importance of decentralised control. In fact Fig. 3 shows the central place of the OS in the system architecture. This can be exploited to receive the detection/error signals generated by both hardware and software protection and handle them if the local hardening devices are not able to do it by themselves.

The following paragraphs (4.2 to 4.4) describe the protections implemented in the system that are then harmonised following our protection scheme. We can find 3

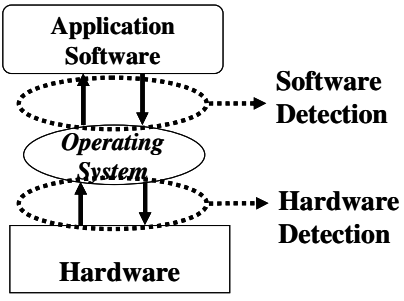


Fig. 3. General Protection Scheme. The Operating System's central position is exploited to make it the arbiter of fault tolerance exploiting detections implemented either in software or in hardware.

types of errors depending on the handling strategy: locally-handled, centrally handled or mixed. Locally-handled errors can be directly and easily corrected by the local hardening devices, like for example SEUs inside the cache memories or SETs in the execution pipeline (see 4.2). Centrally handled errors are errors that cannot be resolved locally and need to call up higher-level constructs to handle them. An example can be SEUs in the pipeline registers or SETs in cache controllers or in the detection-only pipeline (again 4.2) that can be used as trigger for a Context-Switch based rollback (see 4.3). Lastly mixed errors are errors that even if resolvable locally can be passed on to higher-level constructs for further treatment. An example are errors inside the cryptographic IP: even if the potential attack can be hindered by locally resetting the IP it could be useful to pass the detection information to the OS to, for example, erase other sensitive information from the memory or generating an alarm.

4.2 Leon Hardening

In a processor usually the more sensitive units to assure correct operations are the Integer Unit (IU) and the caches. As Fig. 2 shows they are the central parts of Leon, present regardless of the chosen configuration: from now on we will refer to the Integer Unit and the cache system as the "processing unit" to highlight that role. Their impact on performances and execution correctness varies with the application, but techniques tested on them can be easily applied to other components. Moreover they represent two different entity types: the first one (the cache) with little logic and a lot of memory and therefore greatly affected by SEUs, the latter (the IU) mainly composed by logic, more touched by SETs, with the exception of the internal registers. For what concerns protection techniques as a guideline we search for low-cost, low-impact solutions, avoiding extensive replication like TMR, concentrating on protection codes like parity, simple but representative.

The internal registers of each unit can be affected by SEUs: that is the reason why they are separately protected. A parity bit is calculated at writing time and checked at each cycle in parallel of normal computation, generating a validation signal. Being that calculation done in parallel with normal computation the critical path is usually

Table 3. Synthesis results on Virtex II for cache memories and cache controllers hardening

	Slices		4 - LUTs	
Normal	833	--	1455	**--**
SEU Detection	833	(0%)	1507	(+3.6%)
SEU and SET Detection	918	(+10.2%)	1622	(+11.5%)
SEU Recovery	836	(+0.4%)	1511	(+3.8%)
SEU Recovery / SET Detection	940	(+12.8%)	1660	(+14.1%)

not impacted, while the overhead is limited to some flip-flops to store parity and a little simple logic to calculate the parity (in double rail): that technique has been applied to the internal registers of the pipeline, obtaining SEU protection as shown in Table 3.

The other important SEU-prone components are cache memories. Their size imposes a RAM-based implementation, so a slightly different approach is needed. We chose to replicate the protection of Leon-FT proposed in [6] and taken up for [7]: each cache word is protected by two parity bits (one for the tag field, the other for the data), calculated at write time and checked at each read operation. A direct recovery is possible by generating a cache miss and reloading the faulty data from the central memory, supposedly free from the same transient fault. This assumption holds becaude a write-through policy is used, having therefore the central memory always up-to-date. As Table 3 shows that this approach is especially effective for FPGA-based implementations, where cache memories are mapped on internal RAMs: usually these blocks are not completely exploited, leaving some unused bits that can be directly used to store parity without any overhead.

The technique we adopted to protect the pipeline againts SETs is parity prediction, presented in [10] and extended in [11]: The idea is to add a parallel prediction component that just gives the prediction of the parity bit P_{Pre}. If L is the logic to protect, to be efficient the prediction logic L_P should be much simpler in terms of both complexity (otherwise a direct duplication would be better) and speed (not to impact the critical path). Anyway it is I general quite difficult, if not impossible, to manually design L_P starting from L, so a different strategy, illustrated in Fig. 4, is needed. A replica L' is created and its outputs are fed into the parity coder C (a XOR tree), obtaining the predicted parity bit P_{Pre}. L_P is then composed of L'+C and the synthesis tool will simplify it. To verify the code we will simply have to calculate P from L and compare it with P_{Pre}: the checker would better be self-checking, for example in double rail. As can be seen in Fig. 4, some logic is still added, so we have to expect some frequency losses, quite moderated if L_P is well optimized.

Tolerance can be easily achieved if one notices that almost all inputs/outputs are tied to internal registers. Fig. 5 shows the principle: in case of error the update of the registers is not validated and so at next clock cycle the logic will redo the same calculation, correcting the transient error.

That solution is a little more costly both in terms of frequency (outputs must wait error calculation before being validated) and of hardware (I/O must be buffered as they are not already saved in the internal registers). Anyway the overheads should remain much lower than with traditional replications.

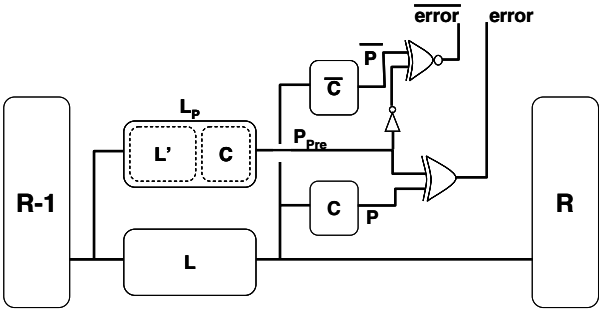


Fig. 4. Detection by parity prediction: the predicted parity bit P_{Pre} is obtained by the unit L_P=replica L'+C. The synthesis tool should be able to greatly simplify it, obtaining a much smaller overhead than with direct duplication.

Table 4. Synthesis results on Virtex II for IU hardening (with cache system)

	Slices	Slice Flip-Flops	4 – LUTs	Clk (MHz)
Normal	2812	1258	4831	46.62
SET Detect	4696 (+67%)	1264 (+0.5%)	8319 (+72%)	45.99 (-1.4%)
SEU and SET Detect	4754 (+69%)	1289 (+2.5%)	8419 (+74%)	45.64 (-2.1%)
SEU Detect and SET Recovery	5239 (+86%)	1824 (+45%)	9274 (+92%)	25.50 (-45%)

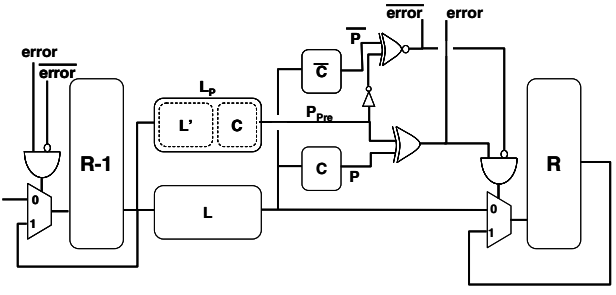


Fig. 5. Recovery by Parity Prediction: values in the registers are updated only if parity coding is respected, otherwise they keep the old value and calculation is done again, correcting the transient error

This strategy has been manually applied to the pipeline, completely protecting it from SETs (parity prediction in the pipeline) and SEUs (parity check for internal registers), as can be seen in Table 4. It must be noted that synthesis is not done on the IU alone but on the higher-level entity: it is the only way to take into account the I/O buffers needed for recovery (see Fig. 5). This means that other little components are

included too (notably the cache system), but they are kept constant in all configurations and their impact on overall results is not that noticeable.

The work done for the IU allowed us to define an automated algorithm to add parity prediction protection to any given entity that we successfully applied to the cache controllers (data and instruction), thus completing the protection of the cache system shown in with SET detection (parity prediction in the cache controllers).

The last protected element that is the general registers bank (not indicated in Fig. 2, but present in the more detailed Fig. 5). It simply is an array of memory elements, usually implemented directly as dual-port RAMs (the high number of 32-bit registers needed by the SPARC architecture makes flip-flop implementation infeasible). That means that they can be easily protected with the same strategy used for caches (see previous paragraph), so their protection does not add any new element to our analysis but simply improve coverage against SEUs.

4.3 OS-Level Hardening

The OS implements Checkpoint and Rollback (C&R) features for the tasks it runs following the scheme proposed in [12] and [13].

The base idea is to reuse the information stored by the OS at context-switch time (the operation of changing the task being executed by the system to achieve multitasking) to resume execution in case of error, as can be seen in Fig. 6. The state stored is usually incomplete, but it is possible to define clear application criteria (see above-mentioned articles for a complete analysis).

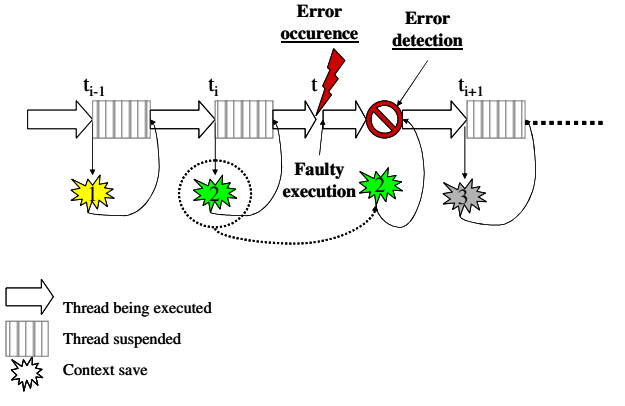


Fig. 6. Rollback to last context switch: information stored by the OS at context-switch time are also used to recover from an error, under some application hypotheses

The result is a light, low-cost protection that enables recovery of any generic transient error detected during task execution. The hardware hardening operations detailed in the section above provide the detection signals that can be used to activate this scheme.

4.4 Cryptographic IP Hardening

The cryptographic IP is hardened to prevent fault-injection attacks. Once more we focus the attention on SETs and SEUs, adequately protecting the circuit.

SEUs may affect the sequential parts: the internal registers (noticeably the one containing the secret key) and the S-Boxes used to compute rounds (see [14] for details on the AES algorithm). Like for Leon caches each byte of the S-Boxes (implemented either in RAM or in ROM) is protected with a parity bit checked at each read operation. The secret key is protected in the same way, adding a parity bit for each byte, thus detecting SEUs changing its value.

Table 5. Synthesis Results for hardened AES IP on Virtex-II, S-Boxes implemented in ROM

Key Size (bits)	Clock Cycles (Coding)	Clock Cycles (Decoding)	Max Clock (Mhz)	Hardware complexity	
				FG	RAMs
128	140	208	37.85	11136	0
192	156	178	40.27	12880	0
256	190	236	38.20	13761	0
all	***	***	39.63	15610	0

***: depends on key size, equivalent to precedent results

Table 6. Synthesis Results for hardened AES IP on Virtex-II, S-Boxes implemented in RAM

Key Size (bits)	Clock Cycles (Coding)	Clock Cycles (Decoding)	Max Clock (Mhz)	Hardware complexity	
				FG	RAMs
128	636	698	42.94	6874	16
192	652	688	26,07	8542	16
256	688	748	43.76	9472	16
All	***	***	38.01	11321	16

***: depends on key size, equivalent to precedent results

SETs affecting the combinatorial logic are detected by timing redundancy. At the end of each round an inverse calculation is done: if the result is different from the original one an error is generated. The same principle is applied for the generation of the new key at each round. The good performances offered by the base IP (down to 70-100 clock cycles for a key size varying from 128 to 256 bits in ROM-style implementation) assure that the hardened IP's performance will be more than

satisfying. Table 5 and Table 6 show the synthesis results for the hardened IP. Once the error has been detected the IP can either directly handle it by resetting itself (and stopping the possible intrusion) or generate an error signal toward the processor that will decide what to do. The situation is quite similar than before synthesis (see Table 1 and Table 2), with the ROM version having better performances in terms of cycles per operation but needing more area and slower frequency than the RAM-based one. It must nevertheless be noted how that timing gap between the two choices is shallower, reduced from some 10 Mhz to 1-2: the RAM implementation pays an heavier toll in terms of speed.

5 Robustness Level

In this paragraph we will evaluate the good robustness level reached by the system presented in section 3 thanks to the hardening interventions detailed in section 4.

Taking as a reference Fig. 1 we can consider the AES block completely protected from transient faults, thanks to the characteristics of timing redundancy: it is almost impossible for the two runs to generate compensating errors. On the other hand some attention should be put on the state machine controlling it: faults causing an illegal jump to the last state could cause the validation of an illegal operation. It is the only situation that could cause an undetected error, as other unwanted state transitions would eventually cause wrong calculation results.

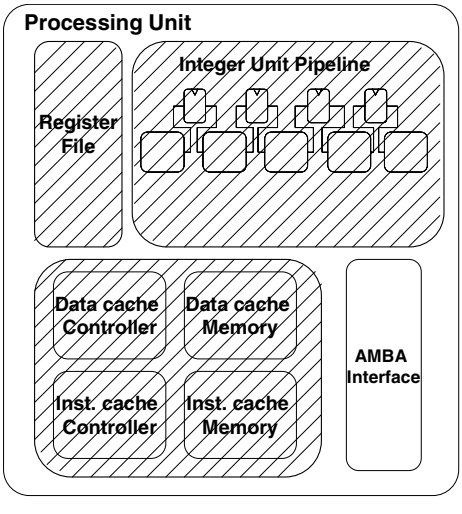


Fig. 7. Hardened components in the Leon processor. As can be clearly seen all components inside the Procesing Unit have been protected with our technique.

The Leon block is protected in its most important parts (see section 3.2): Fig. 7 shows a schema of the processor with the hardened components rayed. As can be easily seen the core of the Processing Unit is completely protected: the integer unit, the cache memories, the cache controllers and the register file are protected from

SETs and SEUs. The limitations of a parity prediction scheme for combinatorial blocks is of course its ability to detect a transient fault only if it generates an odd number of bit errors on the outputs of the block. This can be ensured by using specific synthesis procedures [15], along with careful Placement&Routing . All operations inside the core are then protected.

On the software side the OS gives Checkpoint and rollback capabilities to the software being run, correcting virtually any detected transient fault.

Lastly the RAM is not directly protected, but it is now possible to directly find on the market SEU-tolerant RAMs, usually exploiting Hamming codes.

6 Future Developments

For what concerns security a secure RSA IP is being developed to be added to the system. With that addition it will be able to handle both symmetric secret-key (AES) and asymmetric public-key (RSA) operations, covering the two most important classes in cryptography algorithms. As an example it will be able to perform a PGP-like transaction: exchange of secret keys through a slow but secure public-key asymmetric RSA transaction to open a faster private-key symmetric transaction through AES (that requires both sides to now the secret key). Such a scheme can be well optimised, for instance, with a reconfiguration scheme like in [4]: each cryptographic IP can be an "ASIC-on-demand" and be loaded into the system only when needed.

For what concerns reliability a feasibility study is being conducted on the use of more complex correcting codes (Hamming, Reed-Solomon) to better protect values inside the register file, vital for the correctness of execution, or the possibility of using the proposed Checkpoint and Rollback scheme (see 4.3) to restore them in.

References

1. A. K. Lenstra, *"Memo on RSA Signature Generation In The Presence of Faults"*, private communication (available from the author), September 28, 1996
2. Gaisler Research Homepage, www.gaisler.com, regularly updated
3. Red Hat, eCos Homepage, http://sources.redhat. com/ecos/, updated on a regular basis.
4. C. Plessl, R. Enzler, H. Walder, J. Beutel, M. Platzner, L. Thiele, *"Reconfigurable Hardware in Wearable Computing Nodes"*, 6th International Symposium on Wearable Computers (ISWC) '02, pp 215-222, IEEE, 2002
5. AMBA Homepage, http://www.arm.com/ products/solutions/AMBAHomePage.html, regularly updated;
6. J. Gaisler, *"A Portable and Fault-Tolerant Microprocessor Based on the SPARC V8 Architecture"*, DSN 2002 pp 409-415, 2002
7. Atmel Rad-Hard Product portal, http://www.atmel. com/products/radhard/, regularly updated.
8. D. Hwang, P. Schaumont, Y. Fan, A. Hodjat, B. Lai, K. Sakiyama, S. Yang, I. Verbauwhede, "*Design Flow for HW/SW Accelleration Transparency in the Thumbpod Secure Embedded System*", 40th Design Automation Conference (DAC 2003), Anahiem, California, 2-6 June 2003

9. C.T. Hager, S.F. Midkiff, "*An analysis of Bluetooth security vulnerabilities*", Wireless Communications and Networking, 2003 (WCNC 2003), vol.3 pp1825- 1831, 16-20 March 2003
10. S.-B. Ko and J. –C. Lo, *"Efficient Realization of Parity Prediction Functions in FPGAs"*, Journal of Electronic Testing Theory and Applications (JETTA), Volume 20 Number 5 pp 489-499, October 2004
11. M. Portolan, R. Leveugle, *"A Highly Flexible Hardened RTL Processor Core Based on LEON"*, to be published at the Eight European Conference on Radiation and Its Effects on Components and Systems (RADECS 05), 19-23 September 2005
12. M. Portolan, R. Leveugle, "*Operating System Function Reuse to Achieve Low-Cost Fault Tolerance*", 10th International On-Line Testing Symposium 2004 (IOLTS 2004), Madeira, Portugal, 12-14 July 2004
13. M. Portolan, R. Leveugle*, "A Context-Switch Based Checkpoint and Rollback Scheme*", XIX Conference on Design of Circuits and Integrated Systems (DCIS 2004), Bordeaux, France, 24-26 November 2004
14. NIST AES Homepage, http://csrc.nist.gov/CryptoToolkit/ tkencryption.html, regularly updated
15. N.A. Touba and E.J. McCluskey, *"Logic Synthesis of Multilevel Circuits with Concurrent Error Detection"*, IEEE Transactions on Computer-Aided Design, Vol. 16, No. 7, pp. 783-789, Jul. 1997

Optimal Multicast Loop Algorithm for Multimedia Traffic Distribution

Yong-Jin Lee[1] and M. Atiquzzaman[2]

[1] Department of Technology Education,
Korea National University of Education,
San 7 Darakri, Chungbuk, 363-791, Korea
lyj@knue.ac.kr
[2] School of Computer Science, University of Oklahoma,
200 Felgar Street, Norman, OK 73019, USA
atiq@ou.edu

Abstract. We have presented an optimal algorithm for minimal cost loop problem (MCLP), which consists of finding a set of minimum cost loops rooted at a source node. In the MCLP, the objective function is to minimize the total link cost. The proposed algorithm is composed of two phases: in the first phase, it generates feasible paths to satisfy the traffic capacity constraint, and finds the exact solution through matching in the second phase. In addition, we have derived several properties of the proposed algorithm. Performance evaluation shows that the proposed algorithm has good efficiency for small network with light traffic. Our proposed algorithm can be applied to find multicast loops for real-time multimedia traffic distribution.

1 Introduction

Current computer networks consist of backbone networks that serve as the major highways to transfer large volumes of communication traffic, and local networks that feed traffic between the backbone node and end-user nodes connected to the backbone. Several researchers have proposed algorithms for discovering the topology of the Internet backbone [1-2]. A local area network (LAN) connected to a backbone node, can be regarded as an end-user node. A local network, therefore, consists of a backbone node and several end-user nodes hanging off a LAN.

In a local network, the total traffic volume of end-user nodes that can be served by a port of the backbone node is limited. So, the local network consists of a backbone node (source node) and several trees or loops that cover all end-user nodes to satisfy the constraints on traffic volume. Topology discovery is required to find all the trees or loops in a local network that satisfies the constraints.

Issues related to topology discovery for local network has been classified into two problems in the literature: capacitated minimum spanning tree problem (CMSTP) [3,4] and minimum cost loop problem (MCLP) [5,6]. The CMSTP finds the best way in the least cost aspect to link end-user nodes to a source node. It determines a set of minimal spanning trees with a capacity constraint. In the MCLP, end-user nodes are linked together by a loop that is connected to a port in the backbone node. The links

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 1099 – 1106, 2005.
© IFIP International Federation for Information Processing 2005

connecting end-user nodes have a finite capacity and can handle a restricted amount of traffic, thus limit the number of end-user nodes that can be served by a single loop. The objective of the design is to form a collection of loops that serve all user nodes with a minimal connection cost.

The objective of this paper is to formulate the MCLP and to develop an exact algorithm to solve the problem. We propose a dynamic programming-based exact algorithm. Our proposed algorithm solves the MCLP in two phases: In the first phase, the algorithm uses dynamic programming to generate feasible solutions to satisfy the traffic capacity constraint. In the second phase, it finds exact solution by applying the matching procedure to the set of the capacitated loops found in the first phase.

Our performance evaluation results demonstrate that the time complexity of our proposed exact algorithm is large when the number of nodes in the local network is large. Since the MCLP is NP-hard [7], it is difficult to find the exact solution for large network in short computing time. We, therefore, suggest using our exact algorithm for small local networks (less than thirty nodes), and heuristic methods can be used for large local networks.

The main contributions of this paper are: (i) formulation of the MCLP, (ii) proposing and evaluating its exact solution, and (iii) determining the threshold in choosing between heuristic methods and the exact algorithm depending on the size of the local network.

The suggested algorithm can be applied to the design of synchronous optical network (SONET) and the finding of multicast loops rooted at a source node in order to transfer the message to end-user nodes in the local network.

The rest of the paper is organized as follows. Section 2 presents the mathematical formulation of the MCLP. Section 3 describes our proposed dynamic programming based algorithm for the MCLP. Section 4 evaluates the computational complexity and execution time of our proposed algorithm. Finally, concluding remarks are given in section 5.

2 Formulation of the MCLP

The MCLP (see Fig. 1) is concerned with finding a set of minimal cost loops that are used form the multicast paths for real time traffic, such as multimedia. The end-user

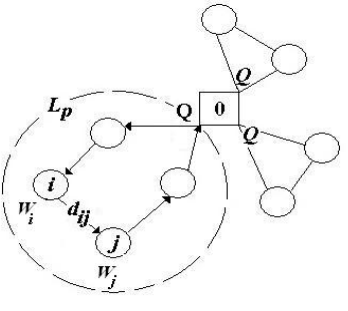


Fig. 1. MCLP

nodes are linked together by a loop connected to a port in the backbone node, where the links connecting the end-user nodes have finite capacity, i.e. they can handle limited amount of traffic (Q). This translates to restricting the number of end-user nodes served by a single loop. The solution of the MCLP results in a collection of loops that serve all end-user nodes with a minimal connection cost.

We now consider the modeling of the MCLP. Assume that there is graph, G=(V, E), where V={0,1,2,..,n}, traffic requirement at node is W_i ($i \in V$-{0}) and link cost between node i and node j is d_{ij} ($j \in V$-{0} and (i,j) $\in E$). Q represents maximum traffic served by single loop. Index 0 represents the source node and can be viewed as an object such as the backbone router with several ports. End-user nodes originate traffics and can be regarded as hosts or switching hubs. The problem formulation for the MCLP is described by Eq. (1). The objective of the MCLP is to find a collection of least-cost loops rooted at node 0. L_p is the p^{th} loop (p=1,2,..,$lcnt$: where $lcnt$ is the number of loops in the solution) whose two of nodes are connected the source node. A particular case occurs when each W_i is equal to one. In that case, the constraint means that no more than Q nodes can belong to any loop of the solution. With this constraint, we can confine the fault to the loop only in which it occurred. x_{ij} represents link between node i and node j (i,j: i=1,2..,n; j=1,2…,n). If link (i,j) is included in any loop (L_p) of the solution, then x_{ij} is set to 1. m represents the number of links in the solution.

$$
\begin{aligned}
&\textit{Minimize} \quad \sum_{i,j} d_{ij} x_{ij} \\
&S.T. \\
&\qquad \sum_{i \in L_p} W_i x_{ij} \le Q \\
&\qquad \sum_{i,j} x_{ij} = m \\
&\qquad x_{ij} = 0 \quad or \quad 1
\end{aligned}
\tag{1}
$$

3 Solution of the MCLP

Having formulated the MCLP in the previous section, we now develop solution for the problem. Our solution consists of two phases: feasible path generation phase and matching phase as described below.

3.1 Feasible Path Generation Phase

To generate the feasible paths for the MCLP using dynamic programming, we define stage variable and state variables: Stage variable, k (k=1,2…), is the number of nodes to form a path rooted at the source node to any arbitrary node j. State variables, j and $S$are the node index to be connected and the set of node indexes included in the path in order to connect node j, respectively. Then, using the principle of optimality, the recurrence relation can be obtained as shown in Eq. (2).

$$
\begin{aligned}
&If \sum_{q \in S} W_q + W_j \le Q \\
&\quad f_k(j,S) = \underset{q \in S, q \neq j}{Min} [f_{k-1}(q, S-\{q\}) + d_{qj}] \\
&\qquad (k = 1,2 \cdots, S \subseteq N_j) \\
&else \\
&\quad f_k(j,S) = \infty
\end{aligned}
\tag{2}
$$

In Eq. (2), $f_k(j,S)$ represents the least cost to connect node j with the paths of which the number of nodes included in S is k to connect node j. $f_k(j,S)$ is set to infinity when the sum of traffics at end-user nodes exceeds Q.

Since boundary condition represents the cost to connect from the source node to node j directly without intermediate nodes, it is defined by Eq. (3).

$$f_0(j,-) = d_{0j} \tag{3}$$

To obtain a feasible solution, we compute $f_1(j,S)$ for all (j,S) satisfying $\sum_{q \in S} W_q + W_j \le Q$ by using f_0. Then, $f_2(j,S)$ is computed using f_1. By repeating this procedure, we reach the phase where $\sum_{q \in S} W_q + W_j > Q$, for all (j,S).

Since $f_k(j,S)$ are infinity for all (j,S) at such a phase, we set L=k. This means that the loop cannot be extended any further. So, paths obtained at the previous stage k (0,1,2,..,L-1) are feasible solutions.

3.2 Matching Phase

At stage k of the feasible path generation phase, $f_k(j,S)$ represents the cost of the path composed of the same elements as $j \cup S$, but the order of elements included in the set is different.

Among $f_k(j,S)$, the minimum cost, $f'_k(P_m)$ is computed. That is, the cost for P_m at stage k, $f'_k(P_m)$ is as follows:

$$
\begin{aligned}
f'_0(P_m) &= f_0(j,-) + d_{j0}, && k = 0 \\
f'_k(P_m) &= \text{Min}[f_k(j,S) + d_{j0}], && k = 1,2,..,\text{L-1} \\
&\forall\ (j,S) \text{ such that } P_m - \{j \cap S\} = \phi
\end{aligned}
\tag{4}
$$

The value of $f'_k(P_m)$ from Eq. (4) represents the cost of the loop including the connection cost to the source node, which is composed of the same node indexes of different order. Node set of the optimal policy, R_m corresponds to $f'_k(P_m)$. R_m is the set of node indexes included in the loop rooted from the source node and represents the node sequence of loop by adding the source node to both end-side indexes. Of course, node 0 is not included in R_m.

Finally, since n nodes have to be included in any loop (R_m) rooted at the source node without duplicate inclusion in the optimal solution, the optimal solution can be

obtained by substituting $f(R_m)$ for $f'_k(P_m)$ from Eq. (4). There can be several collections which have the element, R_m, to satisfy the split condition of set N'. Thus, the global optimal value, F, is the least value among the cost, $f(R_m)$, corresponding to R_m, and can be expressed by Eq. (5).

$$F = \text{Min}\,[\sum f(R_m)\,], \quad \forall\ R_m \text{ such that } \cup R_m = N' \text{ and } R_i \cap R_j = \phi\ \ (i \neq j) \tag{5}$$

3.3 Optimal MCLP Algorithm

From the above model, the optimal MCLP algorithm is described as the following.

OPTIMAL MCLP ALGORITHM

Feasible path generation Phase

1. **for** $k = 0$ and **for all** j such that $j \notin N'$ **do**
2. $\quad f_0(j,-) \leftarrow d_{0j}$
3. **end for**
4. **while** $f_k(j,S) \neq \infty,\ \forall\ (j,S)$ **do**
5. $\quad$ **if** $\sum_{q \in S} W_q + W_j \leq Q$ **then**
6. $\quad\quad$ **for all** k such that $k = 1, 2,.., S$ and $S \subseteq N_j$ **do**
7. $\quad\quad\quad f_k(j,S) \leftarrow \underset{q \in S, q \neq j}{Min}\,[f_{k-1}(q, S - \{q\}) + d_{qj}]$
8. $\quad\quad$ **end for**
9. $\quad$ **else** $f_k(j, S) \leftarrow \infty$
10. $\quad$ **end if**
11. **end while**
12. L $\leftarrow k$

Matching Phase

13. **for** $k = 0$ and **for all** j such that $j \notin N'$ **do**
14. $\quad f'_0(P_j) \leftarrow f_0(j,-) + d_{j0}$
15. **end for**
16. **for all** k such that k=1,2,..,L-1 **do**
17. $\quad\quad m \leftarrow 1$
18. $\quad$ **for all** j such that $j < \underset{q \in S}{\text{Min}}\{\ q\}$ **do**
19. $\quad\quad\quad P_m \leftarrow \{j\} \notin S$
20. $\quad\quad\quad m \leftarrow m+1$
21. $\quad$ **end for**
22. **end for**
23. **for all** k such that k=1,2,..,L-1 **do**
24. $\quad\quad m \leftarrow 1$
25. $\quad$ **for all** (j, S) such that $P_m - \{j \notin S\} = \varnothing$ **do**
26. $\quad\quad\quad f'_k(P_m) \leftarrow \text{Min}\,[f_k(j,S) + d_{j0}]$
27. $\quad\quad\quad m \leftarrow m+1$
28. $\quad$ **end for**

29. **end for**
30. **for all** m **do**
31. **for all** R_m such that $\cup R_m = N'$ and $R_i \cap R_j = \varnothing$ $(i \neq j)$ **do**
32. find $f(R_m)$
33. **end for**
34. **end for**
35. **for all** m **do**
36. $F \leftarrow$ Min $[f(R_m)]$
37. **end for**
38. find the set of optimal loops(R) corresponding to F.

4 Performance Evaluation

Having formulated the MCLP and its solution in Sections 2 and 3, in this section, we evaluate the performance of the algorithm in terms of its computational complexity.

4.1 Properties of MCLP algorithm

We present the following Lemmas in order to show the properties of the proposed algorithm (referred hereafter simply MCLP algorithm).

Lemma 1. Feasible path generation phase ends in the finite stages.

Proof) The finish time of feasible path generation phase depends on the relationship-between $\Sigma^{n}_{i=1} W_i$ and Q. In the worst case ($\Sigma^{n}_{i=1} W_i \approx Q$), we might generate maxi-mumfeasible paths. In such a case, k is close to n. The maximum of k (= L) can be nearly
same as n-1, but more than or equal to n. If L is equal to n, we will find single loop. This is against our assumption ($\Sigma^{n}_{i=1} W_i > Q$). Thus, we can finish the feasible path generation phase in at most n-1 stages.

Lemma 2. The number of additions and comparisons in the feasible path generation phase are $n(n\text{-}1)2^{L}$ and $n(n\text{-}2)2^{L}$, respectively.

Proof) In each stage (k=1,2..,L), we have to add ${}_{n-1}C_k$ for n nodes, and compare ${}_{n-1}C_k$ for the previous stage and n nodes. Therefore, the number of additions $= n\sum_{k=1}^{L} k \cdot {}_{n-1}C_k = n(n-1)\sum_{k=1}^{L} \frac{(n-2)!}{(k-1)!(n-1-k)!} = n(n-1)\sum_{k=1}^{L} {}_{n-2}C_{k-1} = n(n\text{-}1)2^{L}$ and the number of comparisons $= n\sum_{k=1}^{L}(k-1)\cdot {}_{n-1}C_k =$ the number of additions –

$$n\sum_{k=1}^{L} {}_{n-1}C_k = n(n-1)\sum_{k=1}^{L} k \cdot {}_{n-2}C_{k-1} - n\sum_{k=1}^{L} {}_{n-1}C_k \approx n(n\text{-}1)2^{L} - n(2^{L}\text{-}1) = n(n\text{-}2)2^{L}.$$

Lemma 3. The upper bound for the number of additions and comparisons in the matching phase are $n(n\text{-}1)2^{L}$ and $n(n\text{-}2)2^{L}$, respectively.

Proof) In the matching phase, we need not consider the sequence of nodes unlike in the feasible path generation phase. The amount of computations is decreased in most

cases. However, since there might exist the case that the amount of computation is same according to the traffic capacity constraint ($\sum^{n}_{i=1} W_i \leq Q$) regardless of the consideration for node sequence, we can state that the number of additions and comparisons in the matching phase is same as those in the feasible path generation phase in the worst case. Thus, upper bound for the number of additions and comparisons in the matching phase are $n(n\text{-}1)2^L$ and $n(n\text{-}2)2^L$, respectively.

Lemma 4. MCLP algorithm produces the optimal solution.

Proof) In the feasible path generation phase, we enumerate the feasible paths by usingthe optimality principle of dynamic programming. So, there can be no other feasible paths except our solutions. In the matching phase, we first find partitions of which unions compose of the node set, $N' = \{1,2,..,n\}$. Next, we generate loops composed of theabove partitions. These loops are found by adding the index of the source node to indexes of both end nodes included in the partition and are sub-optimal solutions. Then, we select the least cost solution among sub-optimal solutions. Therefore, it is natural for the selected solution to be the global optimal solution.

4.2 Numerical Experiments

We consider the amount of computation in the stage variable (k). It is maximal when the traffic requirement at each node is one ($W_i = 1, \forall i$) and the maximum traffic per single loop is Q. First, for any stage (k), $f_k(j,S)$ must be computed for $k \times_n C_{n-1}$ different (j,S) pairs. Since such computation requires k additions and k-1 comparisons, where k =1 to L, the number of additions and comparisons in the feasible path generation phase are $n(n\text{-}1)2^L$ and $n(n\text{-}2)2^L$, respectively by Lemma 2. In addition, the upper bound for number of additions and comparisons in the feasible path generation phase are $n(n\text{-}1)2^L$ and $n(n\text{-}2)2^L$, respectively by Lemma 3.

Fig. 1 shows the mean real execution time of MCLP algorithm for three different cases (heavy traffc- $Q=1/2\sum^{n}_{i=1} W_i$, medium traffc- $Q=1/3\sum^{n}_{i=1} W_i$, and light traffic- $Q=1/4\sum^{n}_{i=1} W_i$). For each case, ten problems were randomly generated and executed on workstation. MCLP algorithm shows the best efficiency when the sum of the traffics is much less than Q. That is, when the number of nodes is 10, mean execution times for light, medium, and heavy traffic cases are 0.3, 0.3, and 0.2 seconds respectively (since results for 10 nodes are very small, they are not shown in Fig. 1). On the other hand, when the number of nodes is 30, mean execution times for light, medium, and heavy traffic cases are 60, 134 and 200 seconds respectively. As the number of nodes becomes large, memory access time increases sharply. The reason is why $k \times_n C_{n-1}$ storage spaces are required in each stage k to store $f_k(j,S)$. However, the current main memory cannot maintain all the results from the previous computation. If we can use the huge main memory, we will reduce the computation time. In addition, the reason for the less execution time in the light traffics is that since L value in MCLP algorithm becomes small in light traffic case, the amount of computations for MCLP algorithm also becomes small. To summarize, the proposed MCLP algorithm is affected by the traffic volume and Q, and is effective in the case when the number of nodes is less than thirty and the traffic volume is light. Since the execution time increases sharply as the number of nodes is more than thirty, it is desirable to use the heuristic method for the network with large nodes or heavy traffic.

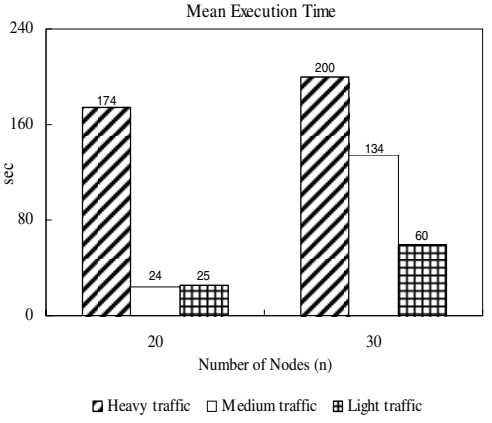


Fig. 2. Mean execution time

5 Conclusions

In this paper, we have presented the problem formulation and an optimal algorithm for the MCLP. The proposed algorithm minimizes the total cost to discover the optimal loops rooted at the source node. It consists of generating the feasible paths using dynamic programming and finding the exact solution by matching procedure. Several properties about the performance of our algorithm were derived, and through experiments, it was shown that the proposed algorithm is effective for small local network of which total traffic volume is relatively smaller than the maximum traffic to be served by a port of the source node. Our proposed algorithm can be used to discover the multicast loops for real-time multimedia traffic. Future work consists of developing a heuristic algorithm applicable to local networks with large number of nodes.

References

1. Breibart, Y., Garofalakis, M., Martin, C., Seshadri, S., and Silberschatz, A.: Topology Discovery in Heterogeneous IP Networks. INFOCOM. (2000) 265-274.
2. Lin, H., Lai, H., and Lai, S.: Automatic Link Layer Topology Discovery of IP Networks. IEEE ICC'99. (1999) 1034-1038.
3. Lee, Y. and Atiquzzaman, M.: Least Cost Multicast Spanning Tree Algorithm for Local Computer Network. IEEE ICCNMC'05. (2005) 268-275.
4. Lee, Y. and Atiquzzaman, M.: Least Cost Heuristic for the Delay-Constrained Capacitated Minimum Spanning Tree Problem. Computer Communications. Vol. 28. (2005) 1371-1379.
5. Gavish, B.: Topological Design of Telecommunication Networks-Local Access Design Methods. Annals of Operations Research, Vol. 33. (1991) 17-71.
6. Lee, Y.: Minimal Cost Heuristic Algorithm for Delay Constrained Loop Network", International Journal of Computer Systems Science &Engineering. Vol. 19, No. 4, CRL Publishing. (2004) 209-219.
7. Lenster, J. and Kan, R.: Complexity of Vehicle Routing and Scheduling Problems. Networks, Vol. 11. (1981) 221-227.

An Effective Method of Fingerprint Classification Combined with AFIS

Ching-Tang Hsieh, Shys-Rong Shyu, and Chia-Shing Hu

Department of Electrical Engineering, Tamkang University, Tamsui, Taipei Hsien, Taiwan 25137, Republic of China

Abstract. In this paper, we present a fast and precise method for fingerprint classification. The proposed method directly extracts the directional information from the thinned image of the fingerprint. We use an octagon mask to search the center point of the region of interest and consider both the direction information and the singular points in the region of interest to classify the fingerprints. In the system, not only is the amount of computation reduced but also can the extracted information be used for identification on AFIS. The system has been tested on the NIST special fingerprint database 4. For the 4000 images in this database, 2000 images are randomly chosen by computer and classified. The classification accuracy reaches 93.1% with no rejection for 4-class classification problem.

Keywords: Biometrics; Fingerprint classification; Wavelets; Poincare index; Singular point detection

1 Introduction

Fingerprints are the innate feature of mankind. The patterns of fingerprints become fixed when a person is about 14 years or older. Due to the invariance and uniqueness of fingerprints, they are used as the tool for crime investigation by the government as far back as A.D. 1897. So far, fingerprints have been used for more than 100 years as the most popular biometric signs of identity. With the large fingerprints database, fingerprint classification is an important subproblem for the automatic fingerprint identification system (AFIS) and automatic fingerprint recognition system (AFRS) [1]. Fingerprint classification improves the computational complexity substantially by reducing the number of candidate matches to be considered, so an effective indexing scheme can be of great help. In this paper we propose a an effective method of fingerprint classification, we extract the only 2D feature vector of the orientation field directly derived from the thinned images. We first use the enhancement algorithm of fingerprint images presented in *Pattern Recognition*, vol. 36 [10]. We compute the orientation following the skeleton of thinned image, so it is more effective without another enhancement for AFIS. Then we use an octagon mask to fast search the center point of the region of interest and extract the feature from the lower half of the octagon mask. Finally, we simplify the feature vectors and use the two-stage classifier to analyze the features and output the result of classification.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 1107 – 1122, 2005.
© IFIP International Federation for Information Processing 2005

The rest of the paper is organized as follows. In Section 2, we briefly review the algorithm of fingerprint enhancement and describe the computation of orientation filed. In Section 3, we detailed search for the centers of the regions of interest using the octagon mask and feature extraction for classification. Then, Section 4 presents our two-stage classification scheme and Section 5 shows the experimental results tested on the NIST-4 database. The conclusions are presented in Section 6.

2 Orientation Field Computation

For the fingerprint classification, the global texture information is more important than the local direction. For an automatic fingerprint identification/recognition system, the algorithm we presented in [10] has shown good results of enhancement, so we combine the fingerprint classification AFIS/AFRS to reduce computation. We describes the processing in the flowing subsections.

2.1 Enhancement Algorithm Review

In an earlier work in *Pattern Recognition*, vol. 36 [10], we propose an effective enhancement method that combines the texture unit of global information with the ridge orientation of local information. Based on the characteristic of hierarchical framework of multiresolution representation, the wavelet transform is used to decompose the fingerprint image into different spatial/frequency sub-images by checking the "approximation region". In considering the global information, all the texture units within an approximation sub-image are processed by textural filters. Based on the hierarchical relationship of each resolution, all other sub-images are processed according to the related texture units within the approximation sub-image. Similarly, the directional compensation is implemented by voting technique in considering the orientation of local neighboring pixels. After these two enhancement processes, the improved fingerprint image is obtained by the reconstruction process of wavelet transform. The flowchart is shown as follows and the results are shown in Fig.2.

2.2 Orientation Field Computation

Because the thinned image has shown the clear characteristic of ridges and valleys on fingerprints, we use a 5×5 mask to compute the orientation following the skeleton of thinned image. The orientation is quantified by the 5×5 mask to 8 directions, S_1 to S_8, as shown in Fig.6, there are 0(π), π/8, π/4,3π/8, π/2,5π/8,3π/4 and 7π/8, respectively. And the pixels of each block belong to the direction D_{p} as

$$D_{\mathrm{p}} = Direction(\max_{i=1,\ldots,8} S_i) \tag{1}$$

and

$$S_i = \sum_{(x,y)\in M} p_i(x,y) \times c(x,y) \tag{2}$$

where M is the 5×5 mask, (x, y) is all the pixels in the mask, $p_i(x,y)$ is the gray value of pixel (x, y) in the ith directional mask and $c(x, y)$ is the gray value of the corresponding pixel (x, y) in the thinned image, respectively. When the computation is completed, each pixel near the skeleton will be given a directional value that they belong to. Fig.3 shows the examples of the processing. Then we divide the image into (height/5) × (width/5) blocks and assign a directional value D_b to the block by

$$D_b = Direction\left(\frac{\sum_{(x,y)\in block} Direction(x,y)}{N}\right) \tag{3}$$

where N is the number of the pixels with direction values in the block.

At this stage, the orientation field has been estimated but is still noisy, we need to smooth it and interpolate values for the noisy region. We use a 5×5 mask to smooth and

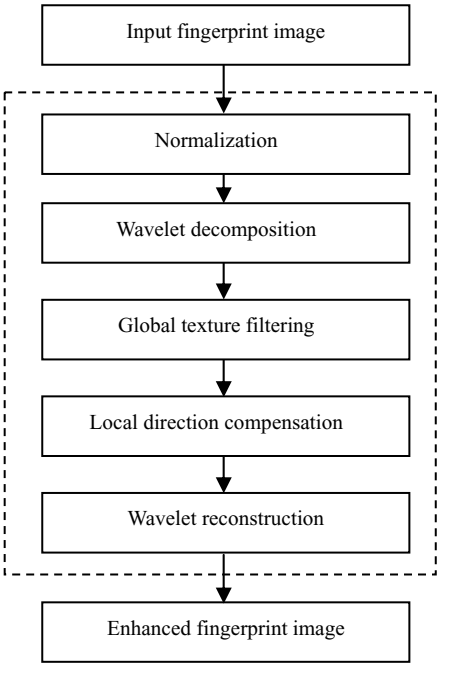


Fig. 1. The flowchart of fingerprints enhancement on identification system (AFIS) in [10]

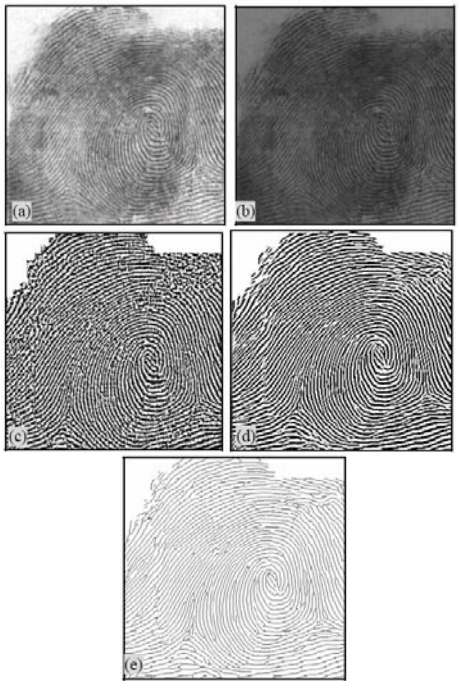


Fig. 2. Results of the enhancement, (a) original fingerprint image, (b) normalized image, (c) the result of global texture unit filtering, (d) the result of local directional compensation, (e) thinned image

interpolate the field by replacing the directional value of center point with the directional value that appears in the mask for most times. After the smoothing, the orientation field is ready for the extraction of the feature for classifying. The result is shown in Fig.4. Figure 5 shows the comparison between the original image and the orientation field.

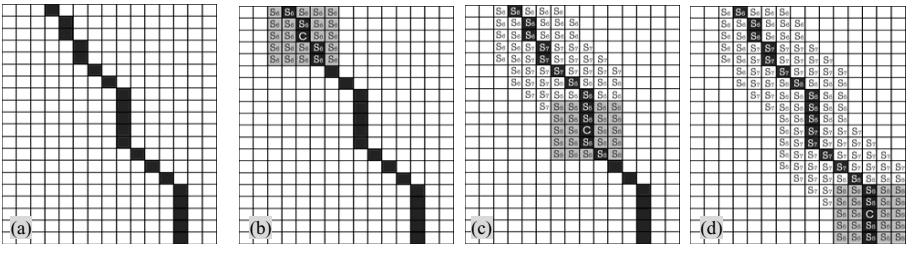


Fig. 3. The example processing of the orientation estimation, (a) the partial skeleton of the thinned image, (b), (c), (d) the sequence of processing

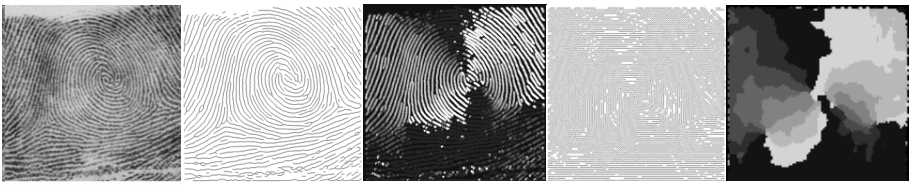


Fig. 4. The result of computation by the 5×5 mask, (a) the original image, (b) the thinned image, (c) the result of the first stage computation, (d) the orientation field, (e) the smoothed orientation field

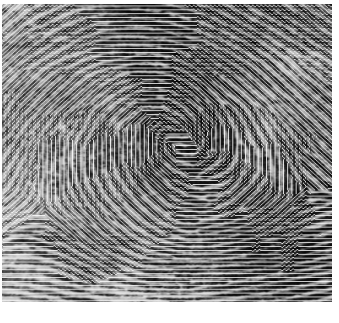


Fig. 5. The comparison between the original image and the orientation field

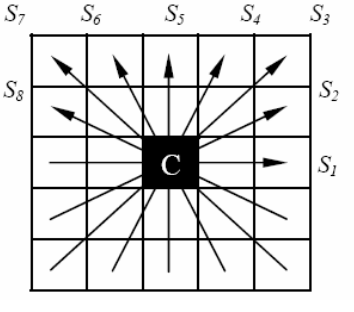


Fig. 6. The 8 directions 5×5 mask represented, C is the center pixel of the mask

3 Feature Extraction

In the stage of feature extraction, the Poincare index, which is derived from continuous curves, is the most popular approach for the singularities based classification [1] [3-5]. However, as mentioned above, the approach is especially hard to deal with the noisy or dim images that lose the information about the regions of singularities. In our method, we find a center of a similar circle from the orientation field and extract the features around the center for classification. We also use the Poincare index but only for detecting that whether the region of interest includes any singular points or not, and the computation will be addressed in a later section. In the next subsection, we first make a discussion on the structures of each class of fingerprints and then describe the detail of the feature extraction.

3.1 The Discussion on the Structures of Each Class

Before extracting the feature, we have to know what feature is helpful to classification and which feature has the most discriminability for fingerprint pattern. So, here we make the discussions about the regular characteristic of each fingerprint class first. We divide the region around the center into four blocks, left upper, left lower, right upper and right lower, and we discuss the directional features on each block for 4-class problem below:

1. Arch (and tented arch): the orientations of texture around the center are symmetrical and the orientations of each block are all downward and outward. (see Fig.6)
2. Left loop: the texture orientations of the two blocks above the center are symmetrical and downward, and the orientations of the two lower blocks are both towards the left. (see Fig.7)
3. Right loop: the texture orientations of the two blocks above the center are symmetrical and downward, and the orientations of the two lower blocks are both towards the right. (see Fig.8)

Whorl: the texture orientations of the left and right blocks are symmetrical each other and downward, and the orientations of the blocks above the center and that are under the center are outwards and inwards, respectively.

By observing the appearances described above, we could draw the conclusion that the orientations of the blocks above the center are consistent in that they are all symmetrical and are downward and outward and the orientations of the blocks under the center have the discriminability of each fingerprint class.

Hence, we use the information above the center to locate the center point of the region of interest and extract the feature from the information under the center point to classify them. The detailed process is described in the next subsection.

3.2 Center Point Location

After we accomplished the computation of the orientation field, we still have to determine the region of interest (ROI) to extract the most representative features for

classification. Locating the center point of the region of interest is an essential step that can influence the accuracy of the classification. We present a fast method to search and locate the center of the region of interest using an octagon mask. In the octagon mask, for the region of each triangle we assign different score with different conditions for the corresponding pixels of the orientation field. As shown in Fig.3, the mask we used is designed by considering the similar structure of a circle and the values 1, 3, 5 and 7 represent the direction S_1, S_3, S_5 and S_7 respectively, as shown in Fig.10. We attempt to find the center of a similar circle from the orientation field of fingerprints, so that the summation of the correlation between the mask and a pattern, if that is an ideal circle, will be the highest. And considering the symmetric of center of fingerprints as mentioned above, we assign different weight for each triangle of the octagon. We increase the weight of upper and both side triangles to emphasize the characteristic of the center point, and the weight of each triangle is shown as Fig.11.

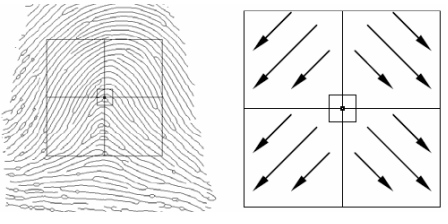


Fig. 6. The orientation characteristic of arch class and its orientations of four blocks around the center

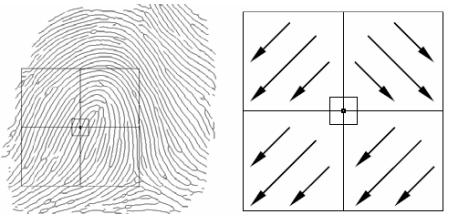


Fig. 7. The orientation characteristic of left loop class and its orientations of four blocks around the center

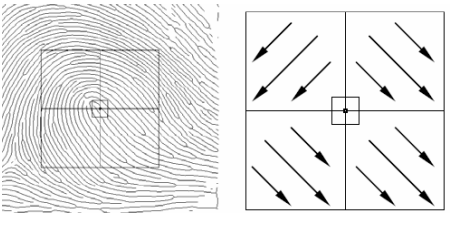


Fig. 8. The orientation characteristic of right loop class and its orientations of four blocks around the center

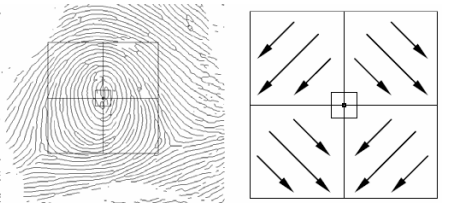


Fig. 9. The orientation characteristic of whorl class and its orientations of four blocks around the center

The center point location algorithm includes following steps:

Step 1: Let D be the orientation field with $D(i, j)$ representing the direction at pixel (i, j).

Step 2: Define an octagon mask M including four directions and eight weights.

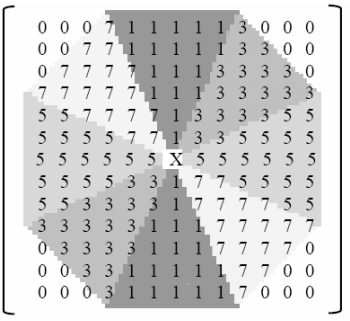


Fig. 10. The mask for capturing the center point; X denotes the evaluated position of possible center point

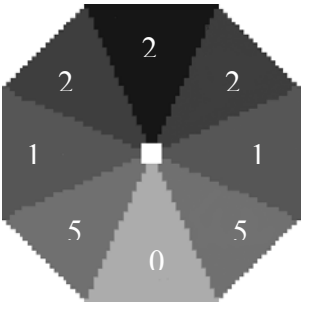


Fig. 11. The weight of each triangle of the octagon

Step 3: Initialize a label image *L* with the same size of *D*. *L* is used to indicate the center point of the region of interest.

Step 4: For each pixel (i_c, j_c) in *D*, define a local region *S* with the same size of *M*. The values of summation $L(i_c, j_c)$ can be estimated by using the following formula:

$$L(i_c, j_c) = \sum_{(i,j)\in M,(i_m,j_m)\in S} \left| dir(i,j) - dir(i_m, j_m) \right| \times weight \tag{4}$$

where $dir(i, j)$ is the direction at pixel (i, j) in *M* and $dir(i_m, j_m)$ is the direction at pixel (i_m, j_m) in *S*. The weight is defined as Fig.11.

Step 5: Search for the minimum value and assign the coordinates as the center point of the region of interest. If there are two equal minimum values in *L,* we will assign the coordinates of the middle point of them as the center point.

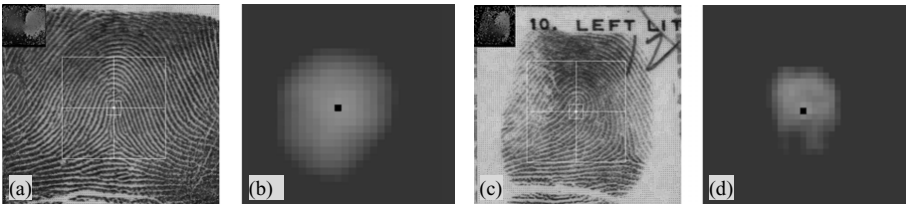


Fig. 12. The results of the center point location, (a) the result of an arch fingerprint, (b) the label image of (a), (c) the result of a left fingerprint, (d) the label image of (c)

3.3 Feature Extraction

As mentioned in section 3.1, the orientation under the center point can most demonstrate the characteristics of each fingerprint class most, so we extract the directional feature of the lower portion of the octagon for classification. Fig.14 shows the region we extract the feature from.

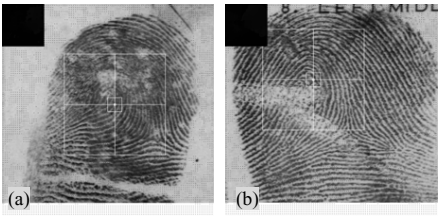


Fig. 13. Some examples of the location on poor quality images, (a) a large number of information is lost, (b) the information of the core is lost

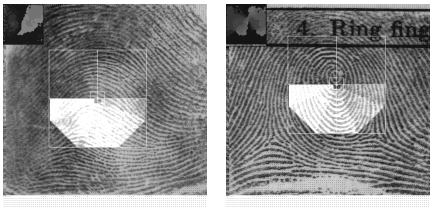


Fig. 14. The regions we extract the features from

We determine the similarities of each direction in triangles to deduce the feature as the input of classification by using the following formulas:

In the left half the *L-score* is estimated by

$$L-score=\sum_{i,j\in L_M} S_{i,j}\times weight \begin{cases} S_{i,j}=10 & \text{when } dir(i,j)=3, \\ S_{i,j}=5 & \text{when } dir(i,j)=2 \text{ or } 4, \\ S_{i,j}=-10 & \text{when } dir(i,j)=7, \\ S_{i,j}=-5 & \text{when } dir(i,j)=6 \text{ or } 8, \\ \text{others,} & S_{i,j}=0, \end{cases} \tag{5}$$

In the right half the *R-score* is estimated by

$$R-score=\sum_{i,j\in R_M} S_{i,j}\times weight \begin{cases} S_{i,j}=-10 & \text{when } dir(i,j)=3, \\ S_{i,j}=-5 & \text{when } dir(i,j)=2 \text{ or } 4, \\ S_{i,j}=10 & \text{when } dir(i,j)=7, \\ S_{i,j}=5 & \text{when } dir(i,j)=6 \text{ or } 8, \\ \text{others,} & S_{i,j}=0, \end{cases} \tag{6}$$

where *dir*(*i*, *j*) is the direction of the corresponding pixel in the orientation field and there are also different weights in each triangle as shown in Fig.15. Finally, we can obtain a 2D feature vector (*L-score*, *R-core*) and then we use the vectors for classification.

4 Classification

Because the derived features describes the directional information on blocks (left lower and right lower), if there are more than one direction in a block, the features will not be able to show the clear feature with some direction. In other words, they will get a low

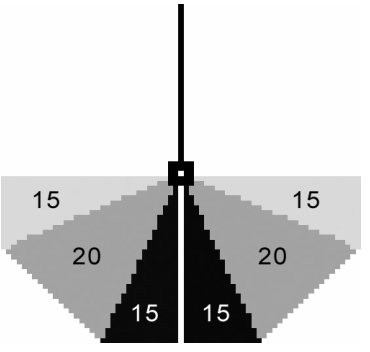


Fig. 15. The weights for feature extraction

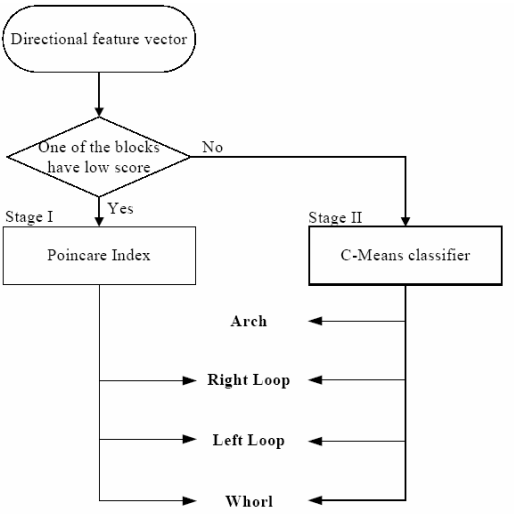


Fig. 16. The flowchart of our fingerprint classification

score on the block that includes the singular points. For this, we detect the singular points in the block with the low score by Poincare index in first stage and classify them into one of three classes of right loop, left loop and whorl. For the others with the clear directional characteristics else, we classify them into one of four classes by c-mean in second stage. The flowchart of classification is shown in Fig.16.

4.1 First Stage: Filtering the Image Including Singularities

We use the Poincare index to filter the images including singular points in the stage. As for fingerprints images, a double care point has a Poincare as 1m a core point 1/2 and a delta point -1/2. The Poincare index at pixel (x, y), which is enclosed by a digital curve, the boundary of the blocks in the paper, can be computed as follows:

$$Poincare(x, y) = \frac{1}{2\pi}\sum_{k=0}^{N_\Psi}\Delta(k) \tag{7}$$

where

$$\Delta(k) = \begin{cases} \delta(k), & if\ |\delta(k)| < \dfrac{\pi}{2}, \\ \pi + \delta(k), & if\ \delta(k) \le -\dfrac{\pi}{2}, \\ \pi - \delta(k), & otherwise, \end{cases} \tag{8}$$

and

$$\delta(k) = \Theta(\Psi_x(i'), \Psi_y(i')) - \Theta(\Psi_x(i), \Psi_y(i)) \tag{9}$$

$$i' = (i+1) \bmod N_{\Psi} \tag{10}$$

Θ is the orientation field, $\Psi_x(i)$ and $\Psi_y(i)$ denote the ith point coordinate (x, y) of the enclosed curve Ψ .

There are three cases that the block probably includes the singular points, it is described as follows:

1. The case of left loop; when the loop feature is not apparent, the directional features are usually unclear due to the fact that some block includes the delta points as shown in Fig.17. Hence if we find that the directional feature towards the left in *L-score* is higher than a predefined threshold and a delta point detected by Poincare index in the right block, it is classified as the class of left loop at the first stage.
2. The case of right loop; the same as left loop, when the loop feature is not apparent, the directional features are usually unclear due to the fact that some block includes the delta points as shown in Fig.18. Hence if we find that the directional feature towards the right in *R-score* is higher than a predefined threshold and a delta point detected by Poincare index in the left block, it is classified as the class of right loop at the first stage.
3. The case of whorl; as shown in Fig.19 and Fig.20, when the texture of a whorl is large, the region of interest we derived in section 3.2 probably includes a singular point. Hence if we find that the directional feature towards the left in *R-score* or towards the right in *L-score* is higher than a predefined threshold and a core point detected by Poincare index in the corresponding block, it is classified as the class of whorl at the first stage. Furthermore, in a few situations there the core point will be replaced by a delta point and we also assign that as the class of whorl as shown in Fig.21.

4.2 Second Stage: Classification

At the second stage, we classify the fingerprint without the first stage by the c-mean classifier. After the second stage classification by c-means classifier, it is all done for

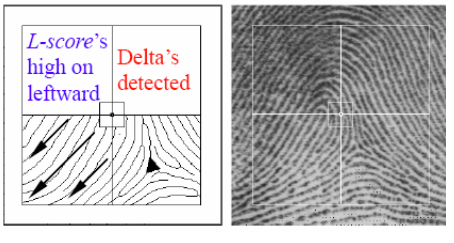


Fig. 17. The left loop includes a delta point in the right block and the *L-score* is high leftward

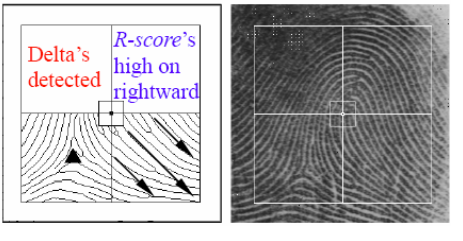


Fig. 18. The right loop includes a delta point in the left block and the *R-score* is high on rightward

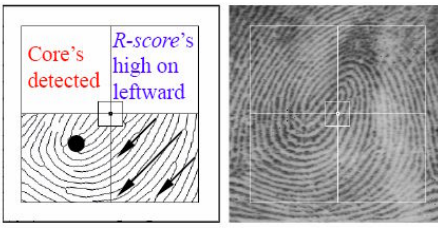


Fig. 19. The whorl includes a core point in the left block and the *R-score* is high on leftward

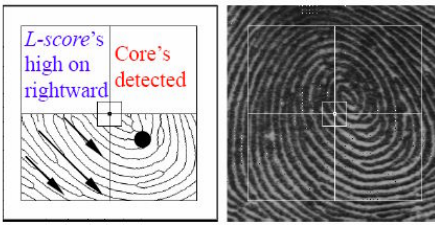


Fig. 20. The whorl includes a core point in the right block and the *L-score* is high on rightward

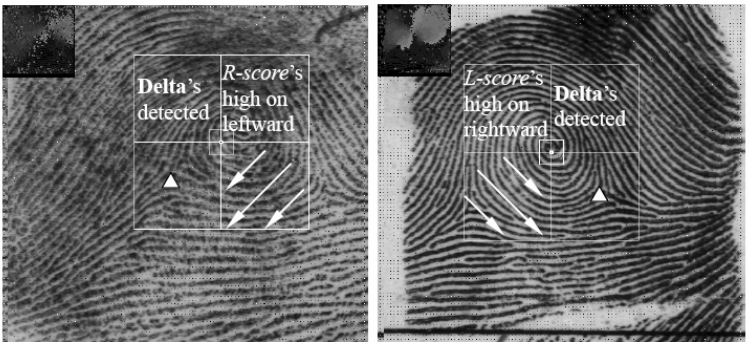


Fig. 21. The situations that the core point is replaced be a delta in the case of whorl

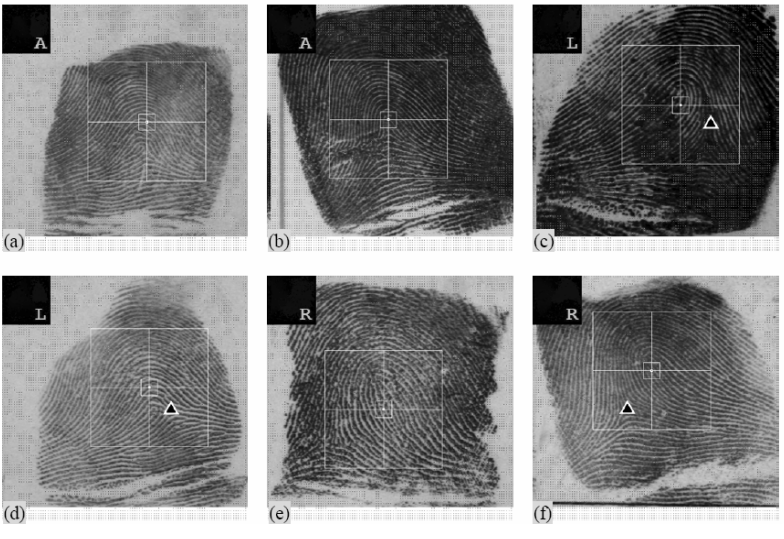


Fig. 22. The results of fingerprint classification, and there are the singular points in (c), (d) and (f)

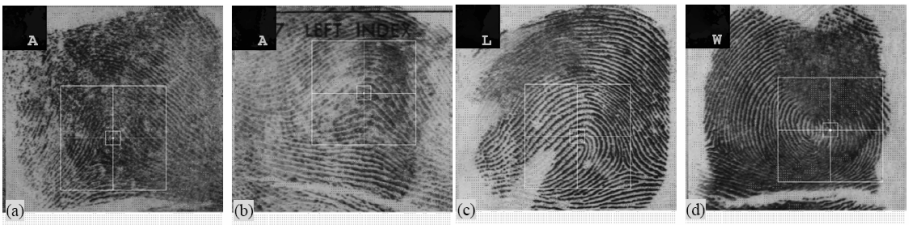


Fig. 23. The results of the images with poor quality

the fingerprint classification. The examples of our results are shown in Fig.22 and that shows the singular points detected at the fist stage such as (c), (d) and (f). And Fig.23shows that the proposed method also can classify the noisy and dim images correctly. The letters A, R, L and W in these figures represent the class of arch, right loop, left loop and whorl, respectively.

5 Experimental Results

In our experiment we use the NIST special fingerprint database 4 (NIST-4) that contains 4000 gray-scale fingerprints images from 2000 fingerprints, and they are distributed evenly over the five classes (arch (A), tented arch (T), left loop (L), right loop (R) and whorl (w)). The classification of each fingerprints are stored in the NIST IHead id field of each file. In our results of the experiment for the 4-class classification problem, the classes A and T are merged into one class and the class index of one image in the NIST-4 database does not necessarily belong to only one class. So we consider our result of the classification to be correct if it matches the one of the class index. This approach has also been adopted by other researchers [1, 3]. We test our algorithm with 2000 fingerprints chosen randomly by computer from NIST-4 data base and the results are shown in Table 1 where the first column shows the class index in the image file header and the fist row labels the assigned class using our method. The table also shows the fail rejection rate (FRR) and the fail accept rate (FAR) in the last column and row, respectively. The accuracy is 93.1% (1862/2000) with zero rejection in our system.

Table 1. Our experimental results for 4-class problem on NIST-4 database with no rejection

True	Exp. as				FRR
	Whorl	Left loop	Right loop	Arch	
Whorl	374	10	6	0	4.1%
Left loop	2	344	3	43	12.5%
Right loop	1	2	354	45	11.9%
Arch	0	16	10	790	3.3%
FAR	0.8%	7.8%	5.1%	10.1%	

Table 2. The experimental results of traditional singular based method for 4-class problem with no rejection

True	Exp. as				Unknown
	Whorl	Left loop	Right loop	Arch	
Whorl	301	10	11	5	63
Left loop	12	334	3	18	25
Right loop	11	5	345	23	18
Arch	9	16	10	760	21

Table 3. The experimental results with pseudo ridge tracing [16] for 4-class problem with no rejection

True	Exp. as				Unknown
	Whorl	Left loop	Right loop	Arch	
Whorl	727	27	26	8	7
Left loop	25	759	5	26	2
Right loop	24	14	740	41	4
Arch	22	38	18	1482	5

From the result table we can see that most of the misclassifications are between the loops and the whorls. The main reason is due to the rotation of loops and the similar pattern between loops and arch, and some of the misclassifications are due to the limitation of the mask size used to extract features; so that the mask cannot include the helpful features like singular points from the images with the lager pattern. Besides, some images have the wrong center of the region of interest located at first stage because of the poor image quality. (See Fig.24)

We also compare the results with the traditional method of singularity based on the same database as shown in Table 2. We can see that the major difference is that the number of the images that cannot be classified by the traditional classifier is up to 127 and the whorl type is up to 63. So the accuracy of the system with singularity based is only 87% for 4-class problem on the same database. There is a complementary texture for the classification of the whorl type using the pseudo ridge tracing presented by Zhang and Yan [16] in recent research and their classification accuracy reaches 92.7% for the 4-class problem on NIST-4 database without rejection as compared in Table 3.

However, there are still many poor quality images, like stained and dim images, on NIST-4 database and they are difficult to recognize even for human. If we discard such obscure images by introducing the rejection scheme, the accuracy of our classification will rise to 93.7% with 1.8 reject rate, 95% with 8.5% reject rate and 96.5% with 19.5% reject rate. Fig.25 shows the corresponding graph of the rejection rate and Fig.26 shows the examples of the rejected images with poor quality.

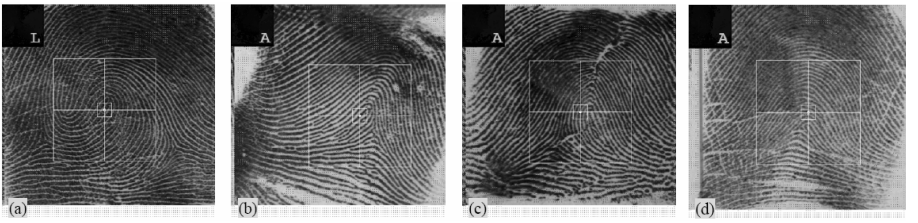


Fig. 24. The misclassified images, (a) a whorl as a left loop, (b) a left loop as an arch, (c) a right loop as an arch, (d) a right loop as an arch

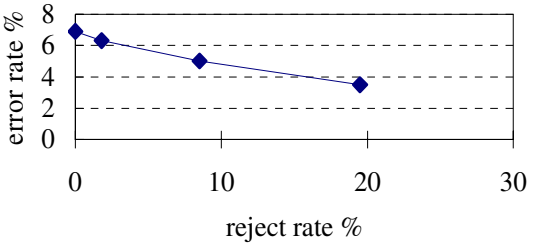


Fig. 25. Four-class error-reject plots of the NIST-4 data base

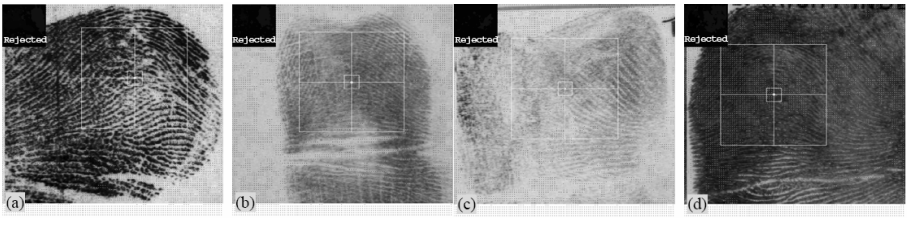


Fig. 26. The examples of the rejected images, (a) the noisy image, (b), (c) the faint images, (d) the over- inked image

Table 4. The computational expense of our system

Procedure	Expense (s)
Image enhancement [10]	0.58
Orientation field computation	0.08
Feature extraction	0.22
Classification	0.00
Total	0.88

Finally, the average computational expense of 1000 fingerprints is recorded at PC Pentium 4 2.6G and the result is shown in Table 4. We can note that the total time is less than one second and the system only spends about 0.3 second on the procedures of the classification. In another word, if the classification is built on an automatic fingerprint identification system, it will have an effective classification only with more than a bit load.

6 Conclusions

We have proposed an effective method for the fingerprint classification, which uses both the orientation characteristic and singular points around the center of the region of interest. As the tables shown above, the results of the system test on NIST-4 database are encouraging. But there is still a lot more that can be done to further improve the results especially on the rotation and the obscure similar pattern.

Acknowledgment

This work was supported by the Nation Science Council, Taiwan, under grant NSC 94-2213-E-032 -020.

Reference

[1] L.C. Jain, U. Halici, I. Hayashi, S.B. Lee, S. Tsutsui, Intelligent Biometric Techniques in Fingerprint and Face Recognition CRC Press, Boca Raton, FL, 1999.
[2] E. R. Henry, *Classification and Uses of Finger Prints*, London: Routledge, 1900.
[3] K. Karu and A.K. Jain, "Fingerprint classification," *Pattern Recognition*, vol. 29, no. 3, pp.389-404, 1996.
[4] B.Meltem, F.A. Sakaraya, and B.L. Evans, "A fingerprint classification technique using directional image," IEEE *Asilomar Conf. on Signals Systems and Computers*, 1998.
[5] M. Kawagoe and A. Tojo, "Fingerprint pattern classification," *Pattern Recognition*, vol. 17, no. 3,pp. 295-303, 1984.
[6] C. L. Wilson, G. T. Candela, and C. I. Watson, "Neural network fingerprint classification," *J. Artificial Neural Networks*, vol. 1, no. 2, pp. 203-228, 1993.
[7] C. I. Watson, and C. L. Wilson, "NIST special database 4, fingerprint database," *Nat'l Inst. of Standards and Technology,* Mar. 1992.
[8] G. T. Candela, P. J. Grother, C. I. Watson, R. A. Wilkinson, and C. L. Wilson, "PCASYS-a pattern-level classification automation system for fingerprints," *Technical Report NISTIR 5647*, Apr. 1995.
[9] A. Senior, "A hidden Markov model fingerprint classifier," *Proc. 31st Asilomar Conf. Singals, Systems and Computers,* pp. 306-310, 1997.
[10] C.T. Hsieh, Eugene Lai and Y.C Wang, "An Effective Algorithm for Fingerprint Image Enhancement based on Wavelet Transform", *Pattern Recognition*, Vol. 36, No. 2, 2003.
[11] I. Daubechies, The wavelet transform, time–frequency localization and signal analysis, *IEEE Trans*. Inf. Theory 36 (1990) 961–1005.

[12] R. W. Zhou, C. Quek and G. S. NG, Novel single-pass thinning algorithm, *Pattern Recognition Letter*, 16(12), 1995, 1267-1275.
[13] Sen Wang, Wei Wei Zhang, Yang Sheng Wang, “Fingerprint Classification by Directional Fields,” Proceedings of the Fourth IEEE International Conference on Multimodal Interfaces. (ICMI’2002).
[14] L. Sha, F. Zhao, and X. Tang, "Improved Fingercode for Filterbank-based Fingerprint Matching," in Proc. of IEEE International Conference on Image Processing (ICIP), Barcelona, Spain, Sept. 2003.
[15] Bogdan Rusyn, Oleg Ostap, “Estimation of Singular Points in Fingerprints Image,” TCSET’2002, Ukraine, Feb. 18-23, 2002, pp.236.
[16] Qinzhi Zhang and Hong Yan, "Fingerprint classification based on extraction and analysis of singularities and pseudo ridge", *Pattern Recognition*, Vol. 37, No. 2, 2004.

A Hierarchical Anonymous Communication Protocol for Sensor Networks

Arjan Durresi[1], Vamsi Paruchuri[1], Mimoza Durresi[2], and Leonard Barolli[2]

[1] Department of Computer Science, Louisiana State University,
298 Coates Hall, Baton Rouge, LA, 70803, USA
{durresi, paruchuri}@csc.lsu.edu
http://www.csc.lsu.edu/∼durresi
[2] Department of Information and Communication Engineering,
Faculty of Information Engineering, Fukuoka Institute of Technology,
3-30-1 Wajiro-Higashi, Higashi-ku, Fukuoka 811-0295, Japan
durresim@franklin.edu, barolli@fit.ac.jp

Abstract. Ensuring anonymity in sensor networks is a major security goal. Using traffic analysis, the attacker can compromise the network functionality by correlating data flow patterns to event locations/active areas. In this paper we present a novel hierarchical anonymous communication protocol that hides the location of nodes and obscure the correlation between event zones and data flow from snooping adversaries. We quantify the anonymity strength of our protocol by introducing a new anonymity metric: Degree of Exposure Index. Our protocol is designed to offer flexible tradeoffs between degree of anonymity and communication-delay overhead.

1 Introduction

Wireless sensor networks, applied to monitoring physical environments, have recently emerged as an important application resulting from the fusion of wireless communications and embedded computing technologies. Sensor networks consist of hundred or thousands of sensor nodes, low power devices equipped with one or more sensors. Potential applications include monitoring remote or inhospitable locations, target tracking in battlefields, disaster relief networks, early fire detection in forests, and environmental monitoring.

With the growth and acceptance of the sensor networks, there has been increased interest in maintaining anonymity in the network. The mere fact that a sensor has sent some information to the base station can reveal extremely important information. For instance, consider a sensor network deployed for intruder detection in which a sensor keeps sensing for intruders. Thus, when an intruder, once in the network area, sees a transmission from a sensor close to his location, can rightly assume that his presence is sensed and might pursue evasive actions immediately. In general, interception of messages containing the physical locations of sensor nodes allows an attacker to locate the nodes and destroy them. The significance of hiding location information from an attacker lies in the fact

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 1123–1132, 2005.
© IFIP International Federation for Information Processing 2005

that the sensor nodes have small dimensions and their physical location cannot be trivially traced. Thus, it is important to hide node locations. Moreover, it should be noted that adversaries can correlate data flow patterns to event locations/active areas using traffic analysis. Therefore, there is a strong need to develop anonymity mechanisms which hide the location of nodes and obscure the correlation between event zones and data flow from snooping adversaries.

Location privacy is a particular case of information privacy and can be defined as the *ability to prevent other parties from learning one's current and past locations* [1]. Anonymity can be defined as the state of being not identifiable within a set of subjects called the anonymity set [2].

Conventional protocols [3],[4] proposed to ensure user anonymity in the Internet are not suitable for sensor networks. To the best of our knowledge, ours is the first protocol that deals with providing complete anonymity in sensor networks.

We present Hierarchical Anonymous Communication Protocol (HACP), a novel protocol that prevents traffic analysis from revealing node information including its location. We use token ring approach for achieving anonymity of communication between cluster heads. Routes are chosen and frames are scheduled to traverse these routes. Each frame is assigned a token and a node can send a message through a frame only if the token is free. Dummy messages are used fora Anonymity inside clusters.

The rest of the paper is organized as follows: Section 2 deals with related work, Section 3 discusses our design goals and network model, Section 4 presents our protocol, Section 5 discusses security and performance results of HACP and Section 6 concludes.

2 Related Work

Anonymous communication for wired networks is a well-studied aspect. A seminal work in the domain of anonymity was notably reported by Chaum in [5]. In [6], Gruteser and Grunwald propose an approach to enhance location privacy in wireless LANs based on disposable interface (MAC) identifiers. The Mist routing project [7] addresses the problem of routing a message to the user while keeping its location private. In [8], Smailagic et al. present two location sensing systems and compare them to the existing location sensing proposals. In [9], Jackson proposes a system that allows user control of the location information disclosure in systems like Active Badge [10]. An important work on IP private roaming has been reported in the framework of the Freedom Network [11], [12]. Recently, Kong and Hong have proposed a protocol for anonymous communication in mobile ad hoc networks [13].

3 Design Goals and Network Model

3.1 Design Goals

We want to design a system that enables anonymous communication. Anonymity is the state of being not identifiable within a set of subjects called the anonymity

set. Here, we define these terms more precisely in the context of hybrid ad hoc networks.

Anonymity is generally classified into source and destination anonymity. Source anonymity is defined as the property that a particular message is not linkable to any source, and vice-versa. A similar definition applies to destination anonymity. Unlinkability in this context means that the probability that a particular message was sent by a given source and/or received by the same destination is the same as imposed by the a priori knowledge. This means that the process of sending and/or receiving messages does not reveal any additional information about the identities of the source and/or destination that was not already known to the attacker prior to the message transmission.

3.2 Network Model

We consider clustered sensor networks because clustering allows for scalability of MAC and routing. Cluster heads also serve as fusion points for aggregation of data, so that the amount of data that is actually transmitted to the base station is reduced. Clustering sensors into groups, so that sensors communicate information only to cluster heads and then the cluster heads communicate the aggregated information to the processing center, may save energy. Many clustering algorithms in various contexts have been proposed [14],[15],[16],[17]. These algorithms aim at generating the minimum number of clusters such that any node in any cluster is at most d hops away from the cluster head.

We use the communication graph G(VCH, E) to represent the network in terms of cluster heads. VCH is the set of cluster heads and E is the set of communication edges (might be paths involving intermediate non-cluster heads) connecting the cluster heads. We assume that G is connected.

We initially fix a spanning tree in the graph. Next, using an Euler tour (that is a DFS tour) of the spanning tree in the graph, we define a ring. Also, the ring formation can use the underlying routing protocol to achieve energy efficiency and load balancing.

We base our protocol on symmetric key cryptographic techniques because of infeasibility of implementation of public key protocols in sensor networks [18]. There exist a number of key pre-distribution schemes for sensor networks to set up secret keys among sensors [19]. We assume that each sensor shares a secret key with its cluster head. Also, each cluster head shares a symmetric key with its neighboring cluster heads in the ring. We use $E(M, K_{ij})$ to represent encryption of message M with K_{ij}, the secret key shared by nodes i and j and $D(M, K_{ij})$ to represent decryption of message M with K_{ij}, the secret key shared by nodes i and j.

Tokens and Frames. At anytime there can be only one frame traversing through the ring. The nodes use a token passing access mechanism to access a frame passing through the network. A node wishing to send data should first receive permission. When it gets control of the token, it may transmit data in that frame. Each frame is of fixed length and contains the status of the token

itself. A token can be either in free status or occupied status. The format of the frame is as follows:

$$< E((Token||E(Message_{Header}, K_{sd})||E(Message_{Data}, K_{sd})), K_{si}) > \quad (1)$$

where K_{si} is the secret key shared between the source node s and node i that is the upstream neighbor of sender s and K_{sd} is the secret key shared between the source node s and destination node d.

The format of the Token is as follows: $< Redundancypredicate||Status >$

Redundancy predicate is used for checking the validity of the frame. For the frame to be verified successfully by node i, upon decryption the *Redundancy predicate* must be fulfilled. *Status* specifies if the token is *occupied* or *free*. If a token is *free*, a node can send data through that frame; else it cannot.

The format of the $Frame_{Header}$ is as follows:

$$< Redundancy\ predicate||Source\ Address||Destination\ Address >$$

The format of $Frame_{Data}$ is as follows: $< Data\ length||Data\ Padding >$

Data length specifies the length of the total data in the packet. This is crucial when the amount of data needed to be sent is not enough to fill the whole frame. In that case, data to be sent is padded with some random number to meet the constraint that the size of the frame is of fixed length.

4 Hierarchical Anonymous Communication Protocol (HACP)

HACP provides two different mechanisms to achieve anonymity - one is based on introducing dummy messages for anonymity with in a cluster and the other is based on ring based approach for anonymous communication with in cluster heads.

4.1 Anonymous Communication with in a Cluster

Inserting dummy traffic in a network is a technique that hides the traffic patterns inside the network, making traffic analysis more difficult [20]. The generation of dummy traffic increases the anonymity of the messages sent through the mix network.

A dummy message is a fake message created by a sensor node. The final destination is its cluster head; the dummy message is discarded by the cluster head. Observers of the network and other nodes cannot distinguish the dummy from a real message.

In HACP, each sensor (including the cluster head) transmits messages at a Poisson rate r_t. Thus, on an average each sensor would send a message every $1/r_t$ seconds. Let r_s denote the sensing rate of each sensor. Thus, whenever there is sensed data to be sent, the sensor encrypts the data message with the secret key

it shares with the cluster head and transmits it. Else the sensor sends dummy messages. Hence, the dummy messages are sent at a rate of $(r_t - r_s)$.

Whenever a cluster head has a message to be sent to one of its cluster nodes, the cluster head simply encrypts the message with the secret key it shares with that sensor and sends. Whenever a sensor senses a packet transmission, it receives the packet and decrypts it with its key and checks if it is a valid packet.

4.2 Anonymous Communication Between Cluster Heads

Whenever a node i receives a frame, it decrypts the frame using the key shared with its downstream node in the ring and verifies the redundancy predicate. Once the *Redundancy predicate* is fulfilled, the following algorithm is executed.

1. If the node has no data to send, it just encrypts the resultant plain frame with the common key shared with its upstream node and retransmits the packet on to the ring.
2. If the status of the token is *free* and the node has some data to send to another node D, then i constructs the frame as follows:
   - Node i constructs $Frame_{Header}$ and $Frame_{Data}$ as explained earlier using key shared with the Destination.
   - Node i sets the *status* field in the token to *occupied*.
   - Computes Equation (1) using its shared key with upstream node and transmits the packet on the ring.
3. If the *status* of the token is set to *occupied*, the node checks if the data in the frame is destined to itself by decrypting $E(Frame_{Header}, K_{sd})$ with the shared key and checking if the *Redundancy predicate* is fulfilled.
   - If the node is able to check the validity of the frame header, then it is addressed to node i, which makes a copy of it. It encrypts the whole frame with the key shared with its upstream node and transmits the frame on to the ring.
   - Else, if the node i is not able to check the validity of the frame header, then it is not the destination and the node just encrypts the whole frame with the key shared with its upstream node and transmits the frame on to the ring.

Once the frame returns to the source, the source repeats the procedure as long as it has data to send. When it has no more data to send it sets the status field of the token to free, assigns the whole frame to some randomly generated data. Then it encrypts the whole frame with its shared key with upstream node and transmits the frame on the ring.

4.3 Multiple Rings

In a network consisting of n nodes, the ring size is n. Thus, a message needs to be transmitted along the whole ring and hence, each message is transmitted n times. To reduce the communication overhead (complexity), we divide the graph into sub-graphs and construct rings with in each sub-graphs. We choose

the same partition mechanism presented in [18]. An example partition is shown in Fig. 1. The dark circle indicates the base station to which all the nodes are communicating with.

Once we have the partition to sub-graphs, we have one ring in each sub-graph, which is formed by an Euler tour on the spanning tree of the sub-graphs. We call the nodes that are part of more than one ring as Junction nodes. There are at most x nodes in each sub-graph, thus the time complexity is no more than x within a sub-graph.

In order to enable communication with node outside a sub-graph, we assign each ring a unique identifier, RID. Also, each node knows the RID of the ring to which the destination belongs. We introduce a new header - $E(Frame_{RID}, K_{sJ})$ - in the frame in order to identify the destination's *RID*, where K_{sJ} is the common key shared by the source with the Junction node that is also part of a ring that has to be traversed to reach the destination. The modified format of the frame as follows: $< E((Token||E(Frame_{RID}, K_{sJ})||E(Frame_{Header}, K_{sd})||$ $E(Message_{Data}, K_{sd})), K_{si}) >$

The format of $Frame_{RID}$ is: $< RIP||RIDD >$. RIP is the redundancy predicate that has to be fulfilled so as to indicate successful decryption. RID_D is the Ring Identifier of the destination's ring. The sender encrypts $Frame_{RID}$ with the key shared with the Junction node that is part of ring that is on the way to the destination's ring.

When a node in one ring has data to send to a node in another ring, then the frame need to be transferred from one ring to another until it reaches the ring of the destination. For this each Junction node maintains a forwarding routing table that specifies the ring a frame addressed to a particular destination ring has to be transferred to. A Junction Node upon successful decryption of $E(Frame_{RID}, K_{sJ})$ stores a copy of the frame and then retransmits the frame. The junction node based on the RID of the destination node, decides to which ring the frame has to be transferred. Then, it waits for a free token on the other ring it has to transmit the copied frame, encrypts the frame with the common key it shares with the next junction node on the way to the destination's ring and transmits the frame. The process continues till the frame reaches the destination's ring, where the Junction node that of RID_D that receives the frame just assigns some random string to $E(Frame_{RID}, K_{sJ})$ and transmits the frame on to the ring RID_D.

This mechanism prevents local traffic from traversing the whole network. Even if an adversary were able to compromise a Junction node, he would just be able to know the ring to which frame was destined to and no more. The attacker could not even figure out the originating ring of the frame. Thus, this mechanism does not reduce the anonymity provided by the protocol. In some situations, only some nodes might have a need for anonymity in which case a ring has to be established only among those nodes. In such cases, the neighbors in a ring need not be physical neighbors in the network and these nodes can communicate using the shortest path available.

5 Performance of HACP

In this section we present the performance of HACP in terms of the overhead imposed and the anonymity provided. Initially, we describe the metrics we would be considering and present the performance of HACP in terms of these metrics.

5.1 Communication Overhead

In HACP, whenever a node has data to send, it captures a free token and sends data in that frame. Else, it just forwards the idle frame. We use the term communication overhead to represent the number of transmissions that correspond to idle frames. It should also be noted that the power consumption of a sensor can be derived from the average current drain [21] given by: $I_{avg} = T_{on} * I_{on} + (1 - T_{on}) * I_{stby}$, where T_{on} is fraction of time receiver or transmitter is on. I_{on} is current drain from battery when receiver or transmitter is on and I_{stby} is current drain from the battery when both transmitter and receiver are off. Therefore, higher the communication overhead higher the T_{on}, which implies higher is the power consumption.

Consider a ring with N number of nodes out of which N_a nodes have data to send at a rate of R packets per unit time. Let us say, a frame can traverse the ring at a maximum of t times in one unit of time. The value of t depends on ring latency, which in turn depends on the transmission time of the frame (T_{tr}), ring traverse time delay (T_t) and processing delay at a node (T_{proc}). Here, we ignore the delay incurred at a node to process the frame before forwarding it. Therefore, $t = \frac{1}{N*T_{tr}+T_{proc}+T_t}$.

If n tokens are present in the ring, then a maximum of $n * t$ frames can be transmitted across the ring. Thus, ideally, we would like to have the following condition satisfied, so that no idle frame is transmitted: $\frac{N_a}{N} * R = n * t$.

Thus, the fraction of idle frames being transmitted over the ring is: $1 - \frac{N_a * R}{N * n * t}$. Thus, communication overhead i.e., number of transmissions corresponding to idle frames, is given by: number of idle frames × number of nodes in the ring $= N - \frac{N_a * R}{n * t}$. The communication overhead in rings for varying sizes and for different number of tokens is presented in Figure 2. The communication overhead increases almost linearly as number of nodes in the ring increases. This behavior

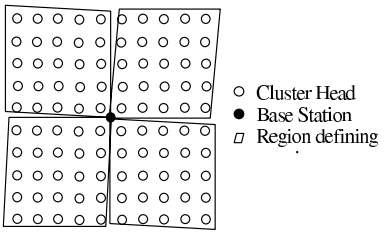


Fig. 1. Partition of a network into multiple rings

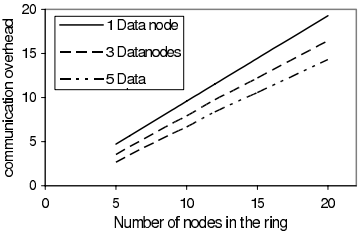


Fig. 2. Communication Overhead Vs number nodes in a ring

is as expected because with more number of nodes in a ring more number of transmissions occur corresponding to each frame generated by any node.

5.2 Data Exposure Index

We introduce a new metric called Data Exposure Index (DEI) defined as follows:

$$DEI = \frac{Number\ of\ data\ generating\ nodes\ on\ the\ ring}{Number\ of\ total\ nodes\ on\ the\ ring} \tag{2}$$

The worst case scenario is when the DEI is equal to one. In this case all nodes on the ring generate data and the attacker's assumption that data is being sent by some node is valid. Less data sources or more the total nodes on the ring reduces the chances of the attacker to identify the data sources.

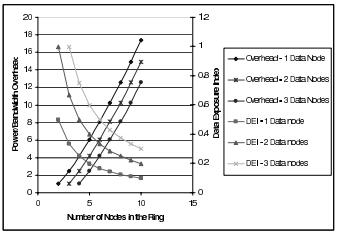


Fig. 3. Tradeoff between Communication overhead and the Data Exposure Index

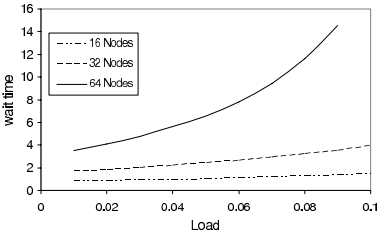


Fig. 4. Wait time in rings of different sizes

Fig. 3 shows the tradeoff between communication overhead and exposure degree. When the total number of nodes on the ring increases, while having the data sources the same, it can be observed that the DEI (right y axis) decreases but the bandwidth/power overhead (left y axis) increases. The user can get different tradeoffs by changing the number of data sources on the ring. For instance, for high anonymity, rings with high number of nodes have to be used, but which results in high communication overhead. Also, to keep the DEI low, ring formation should be such that only few nodes are transmitting at a given point of time. It should be noted most of the related works aim at hiding the communication pattern (i.e., who is talking to who) and not hiding the information if a node is transmitting or not. For these works, the DEI would be one as the attacker would be able to figure out who is transmitting and who is receiving, though he is not able to find out who is receiving from whom.

5.3 Mean Waiting Time

Fig. 4 presents the waiting time calculated as shown in [22] in rings with different number of nodes. As it can be observed, the wait time increases very fast as the

number of nodes in the ring increases. Fig. 5 shows the variation in wait time as the number of active nodes in the ring is varied. As expected, with increase in the number of nodes that have data to send, the wait time increases. From Figs 4 and 5, the tradeoff between number of nodes in the ring and anonymity degree is clear. For time sensitive data which require low latencies, rings with less number of nodes have to be formed which in turn results in less communication overhead but at the same time in less anonymity.

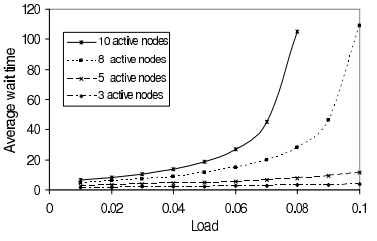


Fig. 5. Wait time vs. number of active nodes in the ring. Total nodes = 32.

6 Conclusion

The data-centric behavior of sensor networks leaves them vulnerable to traffic analysis and identification of event locations and active areas. Therefore, ensuring data anonymity is a crucial research area. We presented Hierarchical Anonymous Communication Protocol (HACP) to achieve anonymous communications in a sensor network. We divide the network into rings and use the concept of tokens and rings to achieve anonymity. We also present the tradeoffs between the overhead imposed and ring sizes. We show that higher anonymity comes at a cost - either higher communication/energy overhead or at higher latency. The choice of the parameters is left to the network administrator and depends on level of security needed and the type of traffic in the network.

References

1. Alastair R. Beresford and Frank Stajano: Location Privacy in Pervasive Computing. IEEE Pervasive Computing, 2(1):46-55, 2003.
2. A. Pfitzmann and M. Kohntopp: Anonymity, unobservability, and pseudonymity - a proposal for terminology. In Proceedings of International Workshop on Design Issues in Anonymity and Unobservability, 2000
3. Goldschlag D., Reed M. and Syverson P.: Onion routing for anonymous and private Internet connections. Communications of the ACM 42, 2 (Feb. 1999), 39-41.
4. Amos Beimel and Shlomi Dolev: Buses for anonymous message delivery. In Second International Conference on FUN with Algorithms, pages 1-13, Elba, Italy, May 2001. Carleton Scientific.
5. D. Chaum: Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM, 4(2), February 1981.

6. M. Gruteser and D. Grunwald: Enhancing location privacy in wireless LAN through disposable interface identifiers: a quantitative analysis. In Proceedings of WMASH, 2003.
7. J. Al-Muhtadi, R. Campbell, A. Kapadia, D. Mickunas, and S. Yi: Routing Through the Mist: Privacy Preserving Communication in Ubiquitous Computing Environments. In Proceedings of the International Conference of Distributed Computing Systems (ICDCS), 2002.
8. A. Smailagic, D. P. Siewiorek, J. Anhalt, D. Kogan, and Y. Wang: Location Sensing and Privacy in a Context Aware Computing Environment. Pervasive Computing, 2001.
9. I. W. Jackson: Anonymous Addresses and Confidentiality of Location. In Proceedings of International Workshop on Information Hiding, 1996.
10. R. Want, A. Hopper, V. Falcao, and J. Gibbons: The active badge location system. ACM Transactions on Information Systems, January 1992
11. http://www.zeroknowledge.com
12. P. Boucher, I. Goldberg, and A. Shostack: Freedom System 2.0 Architecture. Zero-Knowledge Systems Inc. white paper, December 2000.
13. J. Kong and X. Hong: ANODR: ANonymous On Demand Routing with Untraceable Routes for Mobile Ad-hoc Networks. In Proceedings of MobiHoc, 2003.
14. S. Bandyopadhyay and E.J. Coyle: An Energy Efficient Hierarchical Clustering Algorithm for Wireless Sensor Networks: Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003), San Francisco, CA, March 30 - April 3, 2003.
15. C.F. Chiasserini, I. Chlamtac, P. Monti and A. Nucci: Energy Efficient design of Wireless Ad Hoc Networks. In Proceedings of European Wireless, February 2002
16. Ossama Younis and Sonia Fahmy: Distributed Clustering in Ad-hoc Sensor Networks: A Hybrid, Energy-Efficient Approach. Proceedings of IEEE INFOCOM, volume 1, pp. 629-640, March 2004.
17. Ossama Younis and Sonia Fahmy: HEED: A Hybrid, Energy-Efficient, Distributed Clustering Approach for Ad-hoc Sensor Networks. IEEE Transactions on Mobile Computing, volume 3, issue 4, pp. 366-379, Oct-Dec 2004.
18. Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, and Doug Tygar: SPINS: Security Protocols for Sensor Networks. Proceedings of Seventh Annual International Conference on Mobile Computing and Networks MOBICOM 2001, July 2001.
19. 11. Wenliang Du, Jing Deng, Yunghsiang S. Han, Shigang Chen and Pramod Varshney: A Key Management Scheme for Wireless Sensor Networks Using Deployment Knowledge. Proc. of IEEE INFOCOM'04, March 2004.
20. Claudia Daz and Bart Preneel: Reasoning about the Anonymity Provided by Pool Mixes that Generate Dummy Traffic. Appeared in Information Hiding, 6th International Workshop, Lecture Notes in Computer Science, Springer-Verlag, 16 pages, 2004
21. Edgar H. Callaway, Jr., Wireless Sensor Networks: Architectures and Protocols. New York: Auerbach Publications (an imprint of CRC Press). 2003.
22. Alberto Leon-Garcia and Indra Widjaja: Communication Networks: Fundamental Concepts and Key Architectures. McGraw Hill, 2000.

A Network Evaluation for LAN, MAN and WAN Grid Environments

Evgueni Dodonov[1], Rodrigo Fernandes de Mello[1], and Laurence Tianruo Yang[2]

[1] Universidade de São Paulo,
Instituto de Ciências Matemáticas e de Computação,
São Carlos – SP – Brazil
[2] Department of Computer Science, St. Francis Xavier University,
Antigonish, NS, B2G 2W5, Canada

Abstract. The performance of network protocols on different usage scenarios differs significantly, making the protocol choice a difficult question. This had motivated a work that aims to evaluate the TCP, UDP and Sendfile (a POSIX-defined zero-copy TCP access technique) protocols on LAN, MAN and WAN environments, in order to find the most adequate configuration for each protocol. The protocols were evaluated on default configurations, without any application-specific optimizations.

1 Introduction

The increasing usage of grid networks [17] has motivated academic research and commercial projects. Most of these projects are directed to Grid environments for general purpose computing, such as Globus [24], X-Grid [6] and ProGrid [9]. Usually these projects are adopted on widely available hardware and, in most cases, use the default network and system settings.

The nature of the grid networks presumes the existence of a huge number of independent computing resources, each one composed by different processing capacity hardware, and located on several regions, offering different network latency accesses. In this manner, the grid environments must take into account the heterogeneous nature of the environment.

Work have been conduced to evaluate the features of heterogeneous environments and propose new techniques and models for process scheduling [5,12,13,14,22], load transfer [11,18] and network modeling [7,10,23,25]. Some of such work, such as from Mello and Senger [11,13], were developed and applied on grid environments, in order to to evaluate the influences of grid applications on actual networks. This brings up the need to determine when applications should be distributed over grids and at what amplitude. When a distributed application does not involves excessive communication, it is possible to distribute it over networks with non-uniform latencies. Although, when the information exchange is frequent, it is better to allocate nearby computing resources, with low network latencies.

This brings an important subject of how it is possible to determine the nearness of resources and when it is acceptable to distribute applications over distant networks. Based

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 1133–1146, 2005.
© IFIP International Federation for Information Processing 2005

on these needs this paper used the Round-Trip Time (RTT) methodology [19] – which determines the time elapsed to send and receive a message of a fixed size – to evaluate the behavior of some protocols on different network scales. The network evaluated in this work belong to different network types: LAN – a local area network from the University of São Paulo (USP); MAN – a metropolitan area network infrastructure between USP and Federal University of São Carlos (UFSCar); and WAN – a wide area network between USP and St. Francis Xavier University in Canada. Based on these evaluations this paper aims to define good communication rules that should be applied when implementing grid applications. The transfer rate, another evaluation technique, was adopted to analyze the best message size on each network environments.

This paper is divided into the following sections: 2) related work; 3) network evaluations; 4) experiments; 5) conclusions and references.

2 Related Work

Several researches have been conducted in the area of network analysis and evaluation. Most of network evaluations use theoretical models and simulations, with few practical considerations. The most widely used practical network evaluations are performed using measure-based evaluations, such as the Round-Trip Time [19], combined with more specialized and complex methods like the Gaussian Approximation Allocation [8] and Mean-Value Analysis [1].

Besides the measure-based network evaluations, a network can also be described by analytic models [3,4,21]. Among the work related to the analytical network evaluations are the ones by Kherani and Kumar [20], Altman *et al.* [3], Eltetö *et al.* [15], Hockney [19], Iannello *et al.* and Alves and Mello [16].

Kherani and Kumar [20] and Altman *et al.* [3] conduced a behavior evaluation of TCP networks by using analytical models based on two different approaches. In [20], it was introduced a model for network performance prediction based on packet loss probability and network load. The paper analyzes a whole network with all simultaneous connections in order to predict their interactions. The second work [3] used stochastic models for throughput analysis of the random arrival of TCP connections. The authors analyze the network considering the bottleneck probability for each network node. The model is formulated as two equivalent solutions: a fixed point and a nonlinear programming formulation. These work, however, do not consider practical experiments to ensure results.

Eltetö *et al.* [15] analyzed both persistent (ftp-like) and non-persistent (web-like) connections. In this work they introduced a practical analysis framework for TCP networks with three sub-models: the first incorporates a detailed TCP analysis; the second is oriented to network modeling and the third analyzes the dynamics of parallel non-persistent TCP connections. Based on this framework it is possible to estimate the average behavior of network, such as the average queue length and average RTT.

In [19], Hockney proposed a parameterized model to evaluate the elapsed time to transmit messages over networks. On this model one computer, named sender, sends a message of a certain size to another one, named receiver. As the message arrives the

receiver replies it. The time consumed on this send/receive operation is called elapsed time or *round-trip time (RTT)*.

Alves and Mello [16] performed a performance analysis on different network protocols, including TCP, UDP, MPI, PVM and GAMMA protocols. The authors evaluated a LAN (local area network) environment using the Hockney [19] and LogP [2] models. In this work, the protocols were evaluated for different message size (up to 20Kb of data). The RTT, GAP, sending and receiving overhead and the network interface latency values were analyzed.

3 Experimental Environment

This paper presents the evaluation of protocols from the TCP family on different network scales. The evaluations were performed using the Linux operating system on kernel versions 2.6 and 2.4 kernel. For the final experiments, the latest 2.6 family of Linux kernels was considered.Only default protocol configurations were adopted, with no specific protocol optimizations nor tweaking. The evaluated protocols were TCP, UDP and Sendfile over LAN, MAN and WAN networks.

The LAN network evaluation was performed on two Athlon XP 2800+ machines with 1Gb of RAM and Broadcom Gigabit Ethernet cards on a separate switched network.

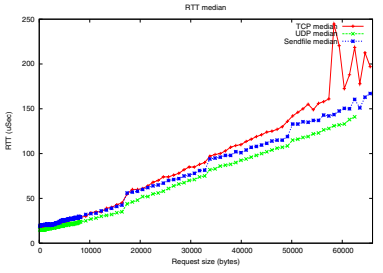
The MAN evaluation was performed using one Athlon XP 2800+ machine with 1Gb of RAM and a Athlon XP 1600+ machine with 256Mb of RAM, on a fast ethernet connection routed by 8 intermediate routers. The physical distance between the machines was approximately $3km$.

Then WAN evaluation was conduced using one Athlon XP 2800+ machine and a Sun UltraSparc cluster composed by five nodes. The Athlon machine was located at ICMC in Brazil, while the Sun machine was located at St. Francis Xavier University in Canada.

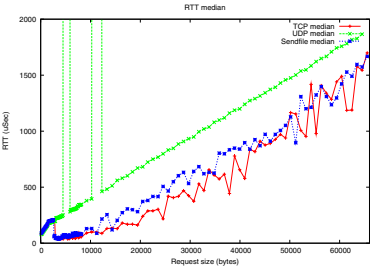
Before the experiments, the expected behaviors of network and directly connected machines were evaluated. The expected network behavior was evaluated using a loopback connection, having all message passing conduced in memory with no physical data transmission. The directly connected network evaluation was performed on one Athlon XP 1600+ machine and one Sempron 2200+ machine, interconnected by a cross-over cable.

4 Network Evaluation

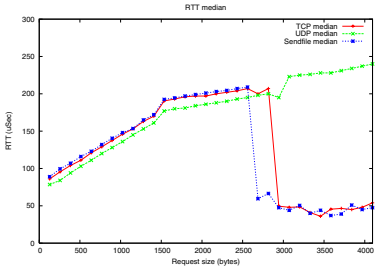
In order to evaluate the network performance for different network classes, a benchmarking application was adopted. This benchmarking application was chosen for the experiments in order to maintain compatible results with previous works [16] Existent performance evaluation applications, such as *iperf* and *netperf*, were not employed as the goal of the experiment was to evaluate the default network and system configurations. The benchmarking application evaluated the time elapsed to transmit and receive equal amount of data, counting the minimum, maximum and medium time taken for



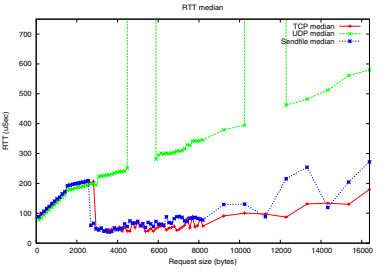
(a) RTT – loopback network evaluation



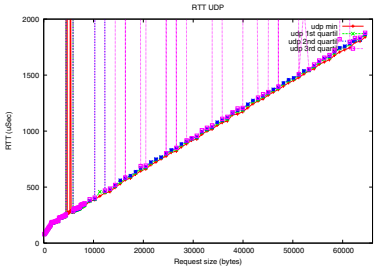
(b) RTT for Gigabit Ethernet LANs



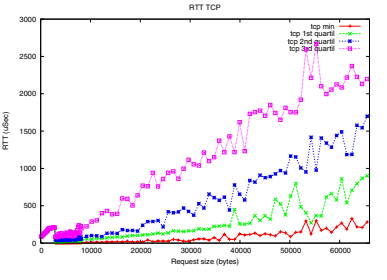
(c) Gigabit LAN – 4KB zoom



(d) Gigabit LAN – 16KB zoom



(e) UDP for LAN networks



(f) TCP for LAN networks

Fig. 1. Evaluation results, part I

this operation among with quartiles distribution [26]. The times distribution was also considered in order to provide a better evaluation of the protocol behaviors. The experiments are shown in the following sections.

4.1 Round-Trip Time Evaluation

Expected protocol behavior evaluation. The expected behavior of the protocols is shown on Fig. 1 (a). This experiment was performed over a loopback connection on a Athlon 2000+ computer running 2.6.10 Linux kernel. In-memory copying was employed for all network operations, having no Internet connection at all. The obtained

median results allowed us to confirm that UDP protocol always get the best times, while all other protocols have a constant behavior during the benchmark. The median values were adopted in order to provide a better visualization of the results, without extremely high and low times that could be very distant from other values.

Local area network using gigabit connection. The intention of the next experiment was to evaluate the protocol behaviors on a Gigabit LAN. The overall performance for all evaluated protocols is shown on Fig. 1 (b). The results are shown using the RTT median response time. It was possible to observe that UDP takes an extremely long time to round-trip a message of sizes around 5 and 11 Kb. Another interesting observation is that the UDP curve is constant, even on cases where TCP and Sendfile present slow behavior, being the upper transfer delay limit.

Another interesting and unexpected detail is the huge performance improvement for TCP-based protocols on cases where message size exceeds the maximum transmission unit (MTU) size of the Ethernet network. As the amount of data exceeds approximately 2800 bytes (excluding the underline protocol overheads), being two times the MTU size, the TCP-based protocols present a significant performance gain. This behavior can be observed more closely on Fig. 1 (c) (zooming the Fig. 1 (b)) where the TCP protocol improves up to 400% when the message size exceeds 2800 bytes. For the Sendfile protocol, this improvement appears earlier, at approximately 2600 bytes. The UDP protocol, on its turn, keeps its elapsed time constantly growing for all message sizes.

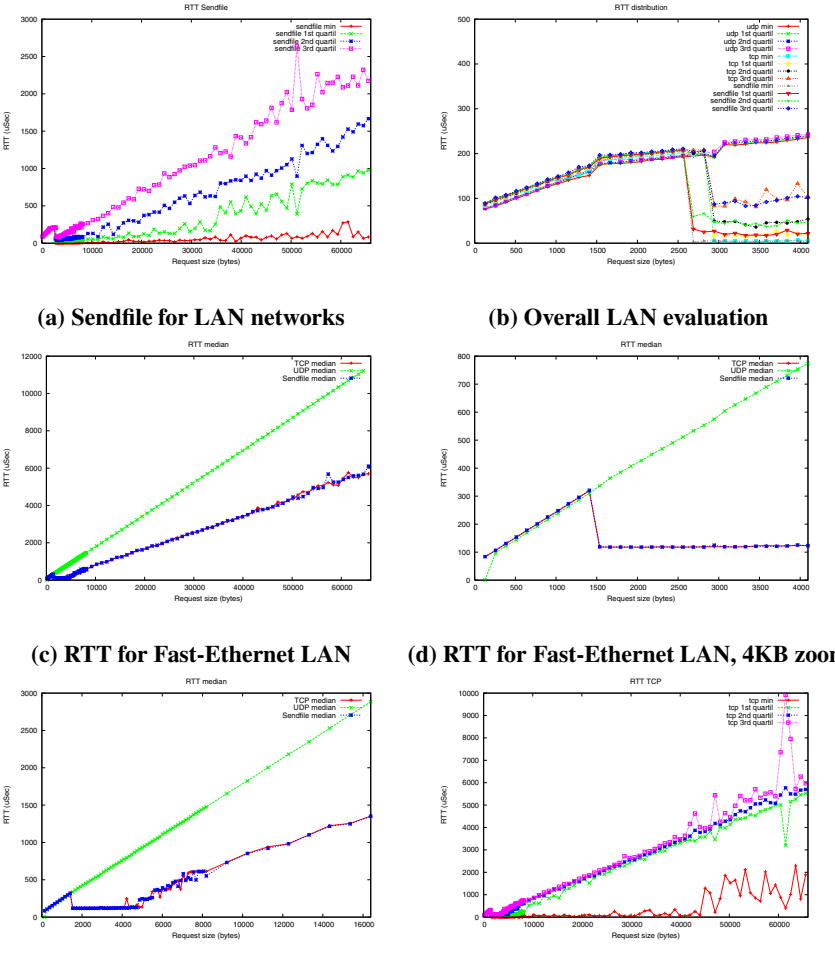
Analyzing further the network behavior for messages of up to 16Kb in size, as shown on Fig. 1 (d), it is possible to observe that the UDP protocol, beside showing a very high latency for some message sizes (messages between 5 and 6Kb in size and messages between 10 and 12Kb in size), manages to obtain the worst results among all evaluated protocols.

On Fig. 1 (e) the first, second and third quartil of RTT for UDP protocol are shown. The quartiles allow a better results visualization, allowing to see the amount of results that happen in up to 25% cases (1st quartil); in up to 50% cases (2nd quartil, or median value) and in up to 75% cases (3rd quartil). Thus, it is possible to see not only the average protocol performance but the overall protocol performance distribution. Based on these quartiles is possible to conclude that, with the exception of some very high latencies for some message, UDP protocol keeps all the time in the same range, with practically no difference between the minimum and maximum times.

The RTT times distribution for TCP protocol (Fig. 1 (f)) allows to conclude that the variance of the values is very high. However, in most cases it is possible to obtain better results than UDP. As can be seen on Fig. 1 (f), only the results of the third quartil are superior to 2000 ms.

The Sendfile protocol is an optimized implementation of the TCP protocol, using in-kernel zero-copy sockets thus requiring less memory copy operations. The performance of this protocol, however, is very similar to the traditional TCP protocol, as shown on Fig. 2 (a), only getting better results for larger messages.

Finally, Fig. 2 (b) shows the behavior of all evaluated protocols for small messages on LAN networks. As can be seen, the behavior of all evaluated protocols is similar until the message size exceeds the MTU of the network. After that, each protocol presents a different behavior.



(a) Sendfile for LAN networks

(b) Overall LAN evaluation

(c) RTT for Fast-Ethernet LAN

(d) RTT for Fast-Ethernet LAN, 4KB zoom

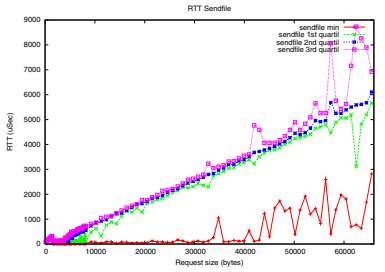
(e) RTT for Fast-Ethernet, 16KB zoom

(f) TCP for Fast-Ethernet networks

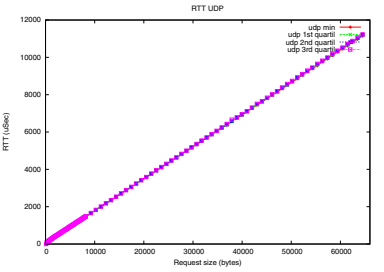
Fig. 2. Evaluation results, part II

Local area network - fast Ethernet cross-over connection. The next experiment was performed in order to evaluate a direct cross-over connection over a Fast-Ethernet network. No switches were used in this benchmark in order to measure the direct connection environment over a Fast-Ethernet network. The overall performance of the protocols is shown on Fig. 2 (c). By observing the figure it is possible to conclude that the TCP-based protocols have a significant performance boost when the message size exceeds 1400 bytes. The UDP protocol keeps the same stable behavior for all message sizes.

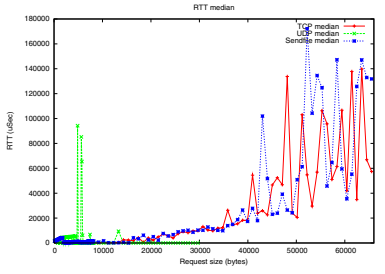
Taking a closer look at the round-trip times, as shown on the Fig. 2 (d), it is clear that the performance gain for TCP-based protocols occurs on messages that have more than



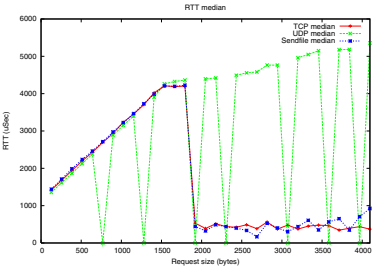
(a) Sendfile for Fast-Ethernet networks



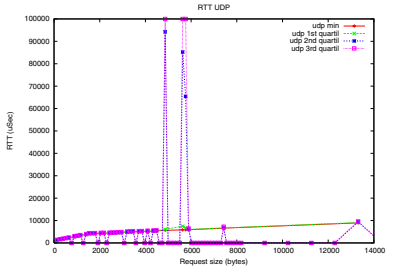
(b) UDP for Fast-Ethernet networks



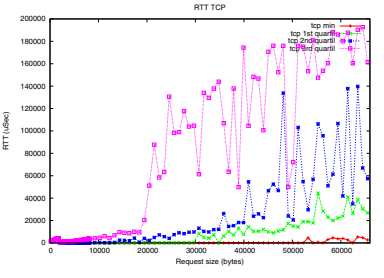
(c) RTT for MAN networks



(d) RTT for MAN networks, 4KB zoom



(e) UDP for MAN networks

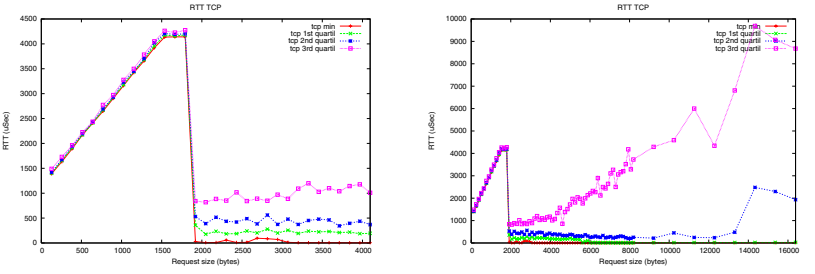


(f) TCP for MAN networks

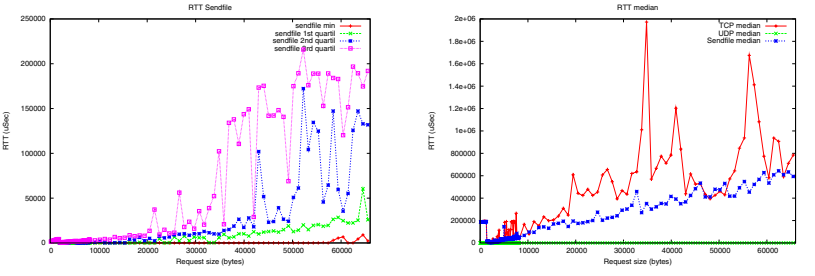
Fig. 3. Evaluation results, part III

1400 bytes in size. Also, as can be seen on the Fig. 2 (e), all evaluated protocols are able to sustain a continuous behavior for small and medium messages, what does not happens on Gigabit Ethernet switched connection network, possible due to the network switch usage.

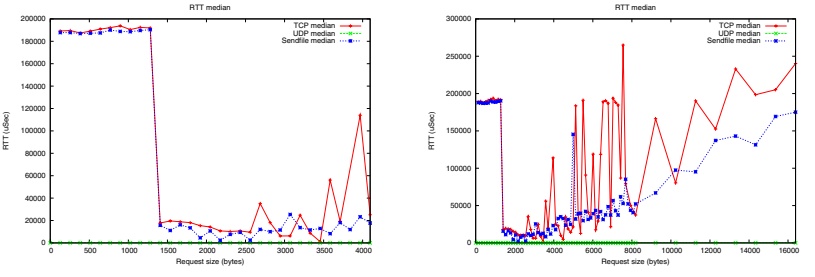
The Fig. 2 (f) presents the minimum, first, second and third quartiles of RTT times for TCP protocol for messages on a larger scale. Analyzing this figure, it is possible to note that the TCP protocol presents a sustained behavior up to messages with 40Kb of data. Exceeding that value, the results present a large variation and instability. The Sendfile protocol offers a similar behavior as shown on Fig. 3 (a).



(a) TCP for MAN networks, 4KB zoom **(b) TCP for MAN networks, 16KB zoom**



(c) Sendfile for MAN Networks **(d) RTT evaluation for WAN networks**



(e) RTT for WAN networks, 4KB zoom **(f) RTT for WAN networks, 16KB zoom**

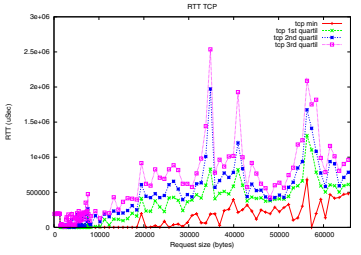
Fig. 4. Evaluation results, part IV

The UDP protocol, on its turn, shows exactly the same behavior for all messages, as shown on Fig. 3 (b).

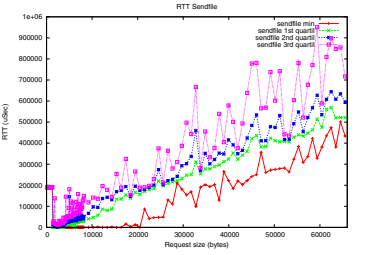
Finally, it was observed that the behavior of this experiment was very close to the expected results, with the exception of TCP-based protocols being actually much faster for larger messages than UDP.

Metropolitan-area network. This experiment evaluated the network RTT times for a wider network. The connection was established from USP to UFSCar, along a three-kilometer 100 Mbit fiber link.

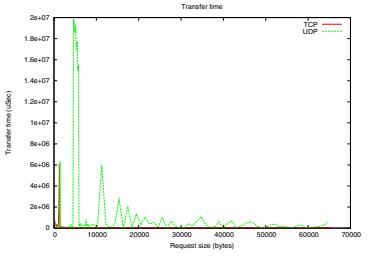
The overall results for this experiment are shown on Fig. 3 (c). As can be observed on the figure, the overall behavior of the protocols remains the same as for the LAN net-



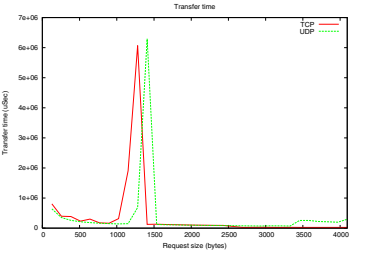
(a) TCP evaluation for WAN Networks



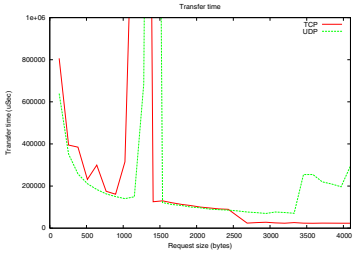
(b) Sendfile evaluation for WAN networks



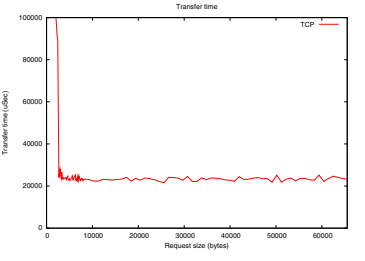
(c) Transfer time for LAN networks



(d) Transfer time for LAN networks, 4KB zoom



(e) Transfer time, 4KB zoom without peaks



(f) Transfer time for LAN networks, TCP protocol

Fig. 5. Evaluation results, part V

work benchmark. However, it was not possible to benchmark the UDP protocol for all message sizes, as when the data size increases to more than 13Kb, the protocol passes to lose too much messages, making impossible the benchmarking without retransmission of the lost messages (which was not the goal of the experiment).

Analyzing the behavior of the protocols for small message sizes up to 4Kb, as shown on Fig. 3 (d), it is possible to ensure that the behavior of the protocols on a MAN network is similar to its LAN equivalent (with the difference that RTT times are much higher than their LAN counterparts). Another interesting detail is that the TCP-based protocols obtain their performance boost on a different message size – while on the

LAN benchmarks this speed up occurred at $2 * MTU$ message size, on the MAN benchmark the TCP protocols got faster for messages of around 1800 bytes. As on the LAN benchmark, the UDP protocol (on Fig. 3 (e)) keeps constant behavior for all request sizes.

While the TCP protocol presented better average RTT times, it also offered much higher times for the messages larger than 20Kb, showing an unpredictable behavior for large messages, as shown on Fig. 3 (f).

Taking a closer look at the TCP protocol for the MAN networks, it was observed that the times distribution is constant for messages under 4Kb of data, improving drastically the performance for messages of around 1800 bytes of data, as shown on Fig. 4 (a). For messages larger than 4Kb of data, the behavior of the protocols starts to vary too much and gets unpredictable, as shown on Fig. 4 (b). Finally, the Sendfile protocol is still very similar to the TCP for the MAN networks, as shown on Fig. 4 (c).

Wide Area Networks. This benchmark was performed in order to evaluate an wide area network between Brazil and Canada.

The overall results for the RTT are depicted on Fig. 4 (d). As can be observed, the UDP protocol was unable to complete the data transfer without packet losses for any message size, thus its results were discarded on this experiment. One interesting detail is the unexpected performance of the Sendfile operation, which managed to get much better than the plain TCP protocol due to the high latency of the network.

Taking a closer look on Fig. 4 (e), it is possible to see that TCP-based protocols maintain the same behavior, once again showing a performance gain for the message size of approximately 1400 bytes.

Evaluating the protocols for messages smaller than 16Kb, as shown on Fig. 4 (f), it is possible to observe that, while the TCP protocol offers an unpredictable behavior, the Sendfile manages to keep a close-to-linear growth for increasing message size, thus offering a much better overall performance than the original TCP protocol.

The unpredictable behavior of the TCP protocol is illustrated on the Fig. 5 (a). Comparing the behavior of the TCP protocol to its Sendfile counterpart, as shown on Fig. 5 (b), it is possible to see that the Sendfile obtains better and more predictable results for wide area networks.

4.2 Transfer Time

Having estimated the network configuration using the RTT experiments, the next step was to determine how these values influence on the network utilization. Thus, another series of experiments was performed. The next experiment consisted in a transfer of a 1Mb file in both direction, varying the message buffer size and measuring the overall transfer time, as illustrated by algorithm on Table 1.

The total time elapsed for a fixed size data transfer can be used for effective transfer rate calculation. As we are using 1MB-size file, the resulting time represents the time necessary to transfer 1MB of data; the transfer rate can be calculated as $rate = 1Mb/time$.

Table 1. Transfer rate evaluation algorithm

```
size := 1024 × 1000;                          // The total transfer size.
msg_size :=< REQUESTSIZE >;                   // Divide into smaller messages.
sent_size := 0;                               // Data already sent.
first_time := get_first_time();               // Get the initial time.
while(sent_size < size) {                     // Do the message transfer.
send(buffer[msg_size]);
receive(buffer[msg_size]);
sent_size := sent_size + msg_size; }
last_time := get_last_time();                 // Get last time.
total_time := last_time − first_time;         // Generate the elapsed time.
```

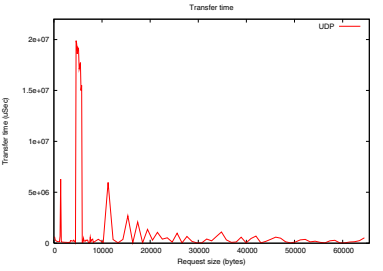
LAN networks. As was observed in the RTT experiments, for LAN networks the network is used in the same manner for any message size with UDP protocol, and obtains the most effective behavior with messages larger than 2800 bytes for TCP protocol.

Fig. 5 (c) shows the global picture for the experiment. As can be observed, the UDP-based data transfer takes a very large amount of time for the data transfer, making the time elapsed by TCP transfer look negligible.

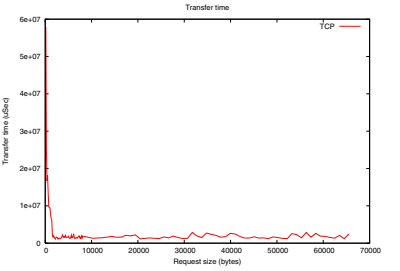
A closer look at the picture, as shown on Fig. 5 (d), shows that a very high elapsed time values takes place for both TCP and UDP protocols for messages with approximately 1Kb in size. A picture without the peak values is shown on Fig. 5 (e). As can be observed, the UDP protocol is actually faster than the TCP for small message size, but when the message size equals to 1500 bytes both protocol show equal speed, and after 2500 bytes of data the TCP protocol gets faster than UDP. Further analyzing the TCP protocol, as shown on Fig. 5 (f), it is possible to see that the elapsed time for data transfer is similar for any message size greater than 2500 bytes. The UDP protocol, as shown on Fig. 6 (a), shows inconstant behavior for all message sizes, with extremely high transfer times in some cases.

MAN networks. In cases of MAN and WAN networks and the UDP protocol it was not possible to complete the 1Mb data transfer without any packet loss, so just the results of the TCP-based protocols are showed. The expected behavior of the TCP protocol is to show a performance boost for messages larger than 1800 bytes in size. The behavior of the TCP protocol is shown on Fig. 6 (b). As expected, the TCP protocol obtains the best performance for messages larger than 1500 bytes. This behavior is better shown on Fig. 6 (c).

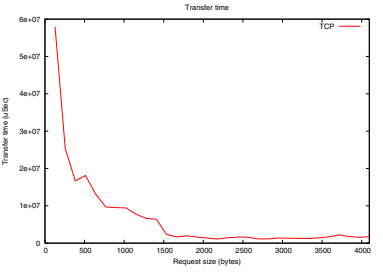
WAN networks. Finally, for the WAN networks the expected behavior was to experience a TCP speed-up for messages larger than 1500 bytes. The real behavior for the WAN networks is shown on Fig. 6 (d). As can be observed, the TCP protocol gets the best performance for messages larger than 1400 bytes in size, as expected. The behavior of the TCP protocol is better showed on Fig. 6 (e).



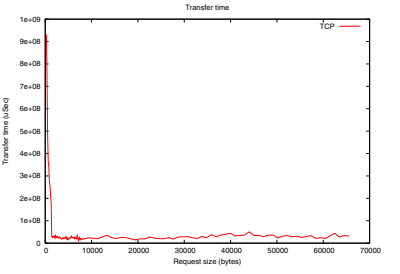
(a) Transfer time for LAN Networks, UDP protocol



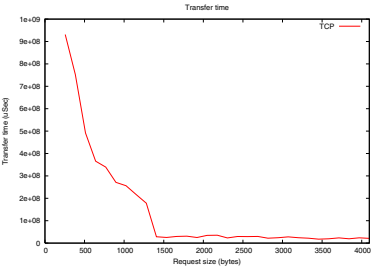
(b) Transfer time for MAN networks



(c) Transfer time for MAN networks, 4KB zoom



(d) Transfer time for WAN networks



(e) Transfer time for WAN networks, 4KB zoom

Message size	LAN	MAN	WAN
0Kb – MTU	UDP	UDP	SF
MTU – 4Kb	SF	SF	SF
4Kb – 16Kb	TCP	TCP	SF
16Kb – 32Kb	TCP	SF	SF
32Kb – 64Kb	TCP	TCP/SF	SF

(f) Summary: Best RTT results

Fig. 6. Evaluation results, part VI

5 Conclusions

This paper performed an evaluation of network protocols for different network configurations, including LAN, MAN and WAN networks. The results were obtained for the Linux OS on its default configuration, and describe both 2.4 and 2.6 Linux kernel families. This paper, however, focused mainly on the currently stable Linux 2.6 family of kernels.

The Fig. 6 (f) resumes the results for the evaluated protocols and network environments.

Concluding the network evaluation of both Round-Time Trip times and data transfer times, it is possible to ensure that for LAN networks, the UDP protocol shows a better

performance for messages smaller than 2800 bytes. When exceeding this message size, the UDP protocol does not offers acceptable performance.

TCP and Sendfile protocols show best performance when the message size exceeds $2 * networkMTU$ bytes for Round-Trip time evaluations. These protocols speedup happen at 2800 bytes for LAN networks, 1800 bytes for MAN networks and 1400 bytes for WAN networks.

UDP protocol is not recommended to be used on MAN or WAN networks due to a slower behavior and excessive packet loss. TCP protocol, on its turn, shows best results when used on LAN networks, being surpassed by the Sendfile for MAN and WAN networks. In all cases TCP-based protocols manage to obtain better transfer rate than the UDP protocol – for example, it is much faster to transfer 4Kb of data using TCP protocol than 1Kb of data using UDP.

The Sendfile protocol offers a better performance due to zero-copy sockets usage and in-kernel operations. This is observed mainly on WAN networks, due to high network latency values. This is very important for Grid environments which are based on long distance connections. Also, Sendfile shows the best performance for messages between approximately 2.8Kb and 4Kb in size in all cases.

References

1. I. F. Akyildiz. Mean value analysis for blocking queueing networks. *IEEE Transactions on Software Engineering*, 14(4):418–428, 1988.
2. A. Alexandrov, M. F. Ionescu, K. E. Schauser, and C. Scheiman. LogGP: Incorporating long messages into the LogP model for parallel computation. *Journal of Parallel and Distributed Computing*, 44(1):71–79, 1997.
3. E. Altman, K. Avrachenkov, and C. Barakat. TCP network calculus: the case of large delay-bandwidth product, 2002.
4. E. Altman, K. Avrachenkov, C. Barakat, and K. Chandrayana. Complementarity formulation for TCP/IP networks: uniqueness of solution and relation with utility optimization.
5. Y. Amir. An opportunity cost approach for job assignment in a scalable computing cluster. *IEEE Transactions on Parallel and Distributed Systems*, 11(7):760–768, July 2000.
6. Apple Computer, Inc. X-grid project. http://www.apple.com/acg/xgrid/.
7. S. Chodnekar, V. Srinivasan, A. S. Vaidya, A. Sivasubramaniam, and C. R. Das. Towards a communication characterization methodology for parallel applications. In *Proceedings of the 3rd IEEE Symposium on High-Performance Computer Architecture (HPCA-97)*, page 310, 1997.
8. H.-W. Chu, D. H. K. Tsang, and T. Yang. Bandwidth allocation for VBR video traffic in ATM networks. In *ICCCN*, page 612, 1995.
9. P. Costa, S. D. Zorzo, and H. C. Guardia. Progrid: A proxy-based architecture for grid operation and management. In *Proceedings of International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD03)*, 2003.
10. D. E. Culler, R. M. Karp, D. A. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. von Eicken. LogP: Towards a realistic model of parallel computation. In *Principles Practice of Parallel Programming*, pages 1–12, 1993.
11. R. F. de Mello and L. J. Senger. A new migration model based on the evaluation of processes load and lifetime on heterogeneous computing environments. In *International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD04)*, page 6, 2004.

12. R. F. de Mello and L. J. Senger. A new slowdown model for heterogeneous multicomputing environments. *submited*, pages 1–23, 2005.
13. R. F. de Mello and L. J. Senger. A routing load balancing policy for grid computing environments. *submited*, pages 1–16, 2005.
14. R. F. de Mello et. al. Analysis on the significant information to update the tables on occupation of resources by using a peer-to-peer protocol. In *16th Annual International Symposium on High Performance Computing Systems and Applications (HPCS-02)*, Moncton, New-Brunswick, Canada, June 2002.
15. T. Eltetö, P. Vaderna, and S. Molnar. Performance analysis of tcp networks loaded by web traffic.
16. E. A. Ferreira and R. F. de Mello. Avaliação de desempenho de protocolos e bibliotecas de comunicação entre processos. In *3rd International Information and Telecommunication Technologies Symposium*, page 8, 2004.
17. I. Foster. Interneting computing and the emerging grid. `www.nature.com/nature/webmatters/grid/grid.html`, 2000.
18. M. Harchol-Balter and A. B. Downey. Exploiting Process Lifetimes Distributions for Dynamic Load Balancing. *ACM Transactions on Computer Systems*, 15(3):253–285, August 1997.
19. R. W. Hockney. The science of computer benchmarking. In *SIAM*, 1996.
20. A. Kherani and A. Kumar. Stochastic models for throughput analysis of randomly arriving elastic ows in the internet, 2002.
21. J. Padhye, V. Firoiu, D. Towsley, and J. Krusoe. Modeling TCP throughput: A simple model and its empirical validation. *Proceedings of the ACM SIGCOMM '98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 303–314, 1998.
22. L. J. Senger, R. F. de Mello, M. J. Santana, and R. H. C. Santana. Improving scheduling decisions by using knowledge about parallelapplications resource usage. *submited*, pages 1–12, 2005.
23. A. Sivasubramaniam. Execution-driven simulators for parallel systems design. In *Winter Simulation Conference*, pages 1021–1028, 1997.
24. The Globus Project. http://www.globus.org.
25. J. S. Vetter and F. Mueller. Communication characteristics of large-scale scientific applications for contemporary cluster architectures. *Journal of Parallel and Distributed Computing*, 63(9):853–865, 2003.
26. W.C.Shefler. *Statistics: Concepts and Applications*. The Benjamin/Cummings, 1988.

SVM Classifier Incorporating Feature Selection Using GA for Spam Detection

Huai-bin Wang[1], Ying Yu[2], and Zhen Liu[3]

[1] Dept. of computer science, Tianjin University of Technology,
300191, Tianjin, China
whb@tengda.net
[2] School of Computer Science and Technology, Harbin Institute of Technology,
150001, Harbin, China
yuying@insun.hit.edu.cn
[3] Nagasaki Institute of Applied Science, Japan
536 Aba-machi, Nagasaki 851-0193, Japan
liuzhen@cc.nias.ac.jp

Abstract. The use of **SVM** (Support Vector Machines) in detecting e-mail as spam or nonspam by incorporating feature selection using GA (Genetic Algorithm) is investigated. An GA approach is adopted to select features that are most favorable to **SVM** classifier, which is named as GA-SVM. Scaling factor is exploited to measure the relevant coefficients of feature to the classification task and is estimated by GA. Heavy-bias operator is introduced in GA to promote sparse in the scaling factors of features. So, feature selection is performed by eliminating irrelevant features whose scaling factor is zero. The experiment results on UCI Spam database show that comparing with original **SVM** classifier, the number of support vector decreases while better classification results are achieved based on GA-SVM.

Keywords: Support Vector Machines (SVM); Genetic Algorithm (GA); Feature selection; Spam detection

1 Introduction

Spam is defined as junk e-mail message that is unwanted delivered by the internet mail service. There are various methods to classify e-mail [1,2,3] .Technical solutions to detect spam include filtering based on sender address or header content. The problem with filtering is that a valid message may be blocked sometimes. Rather, we distinguish whether an e-mail is spam according to features. The selection of features is flexible in deferent scenario, such as percentage of words in the e-mail that match specified word, percentage of words in the e-mail that match specified character, average length of uninterrupted sequences of capital letters etc.

In this paper, **SVM** is utilized to detect spam by incorporating feature selection using GA. As a new approach of pattern recognition, **SVM** [4] is based on Structural Risk Minimization (SRM). It is suitable to deal with magnitude features problems with a given finite amount of training data. Even so, there may be a large number of

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 1147 – 1154, 2005.
© IFIP International Federation for Information Processing 2005

available features to be redundant, noisy or unreliable in the training data to deteriorate the success of **SVM** strongly. So feature selection is necessary to improve the performance of **SVM** for more pleasing results.

The approach for feature selection can be divided into wrappers, filters and embedded methods essentially [5,6]. In this paper, feature selection and **SVM** classifier design are performed simultaneously in the framework of embedded methods, which is named as GA-SVM.

The rest of the paper is organized as follows. Section 2 reviews the **SVM** with feature scaling parameter and related algorithm on how feature relevance scalings are derived from the **SVM** methodology. In Section 3, details of the proposed algorithm are introduced. In section 4, numerical experiments on UCI real-world data show the results of our proposals. And Section 5 is the conclusion arisen from this work.

2 SVMs with Feature Scaling Parameter

Given l independent and identically distributed examples

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_l, y_l), \quad \mathbf{x}_i \in \mathbf{X} \subset R^n, \quad y_i \in Y = \{1, \cdots, k\} \tag{1}$$

The core desired solution of **SVM** is to find optimal hyperplane between two classes which can separate two different classes with the maximal distance, the standard **SVM** require the solution of the following convex Quadratic Programming (QP) optimization problem:

$$\begin{aligned} &\underset{w,b}{Minimize} \quad \Phi(w,b) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\xi_i \\ &subject\ to \quad y_i\left(w^T \varphi(x_i) + b\right) \geq 1 - \xi_i,\ \xi_i \geq 0, i = 1, 2, \cdots, l \end{aligned} \tag{2}$$

$C > 0$ is the penalty parameter to control the trade-off between maximizing the margin and minimizing the training error term ξ_i.

Scaling factor is a discrete real-valued weight that is associated with each feature to measure feature's importance. There are two prospective benefits of this method at least: Feature selection is performed by eliminating irrelevant features whose scaling is zero; an **SVM** classifier that has enhanced generalization ability can be learned concurrently. By introducing scaling factor, above QP problem is written as follow form:

$$\begin{aligned} &\underset{w,b}{Minimize} \quad \Phi(w,b) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\xi_i \\ &subject\ to \quad y_i\left(w^T (\Delta x_i) + b\right) \geq 1 - \xi_i,\ \xi_i \geq 0, i = 1, 2, \cdots, l \end{aligned} \tag{3}$$

in which

$$\Delta = diag\{\delta_1, \cdots, \delta_l, \cdots, \delta_n\}, 0 \leq \delta_i \leq 1, i = 1, \cdots, n \tag{4}$$

Δ is n-dimension diagonal matrix of scaling factors.

J. Weston et al[7] find the optimum value of Δ that can minimize

$\frac{\text{Radius}}{\text{margin}} = R^2(\Delta)\mathbf{w}^2(\Delta)$ or some other differentiable criterion by gradient descent algorithm. In the paper of Olivier Chapelle et al [8], a similar iterative procedure is used in computing preprocessing scaling parameters. Alain Rakotomamonjy [9] has presented different criteria for feature selection algorithms. These criteria are derived from generalization error bounds of the **SVM** theory. Balaji Krishnapuram et al [10] adopt a Bayesian approach to learn both an optimal nonlinear classifier and a subset of predictor features simultaneously that are most relevant to the classification task. In Yves Grandvalet [11], the metric is automatically tuned by the minimization of the standard **SVM** empirical risk, where scale factors are added to the usual set of parameters defining the classifier.

3 Incorporating Feature Selection with SVM Classifier Design Using GA

3.1 Problem Formulation

The basic intuition behind our approach to estimate the feature scaling and design classifier problems jointly is to extend the set of parameters to be learned. Consequently, the function similar to Ref. [10] is considered:

$$f(x,\theta,b) = sign\left[\sum_{i=1}^{m} \lambda_i y_i K_\theta(x_i, x_j) + b\right] \tag{5}$$

in which, some common types of $K_\theta(x_i, x_j)$ are listed as follow:

(1) The polynomial kernel:

$$K_\theta(x_i, x_j) = \left(1 + \sum_{l=1}^{n} \theta_l x_i^{(l)} x_j^{(l)}\right)^d \tag{6}$$

where *d* is the degree of polynomial

(2) The Radial Basis Function (RBF) kernel with parameter *g*:

$$K_\theta(x_i, x_j) = \exp\left(-g\sum_{l=1}^{n} \theta_l \left(x_i^{(l)} - x_j^{(l)}\right)^2\right) \tag{7}$$

(3) The sigmoid kernel (only for some values of coefficients τ_1, τ_2)

$$K_\theta(x_i, x_j) = \tanh\left(\tau_1 \sum_{l=1}^{n} \theta_l x_i^{(l)} x_j^{(l)} + \tau_2\right) \tag{8}$$

In above forms, $x_i, x_j \in R^n$ is *n*-dimensional vector, $x_i^{(l)}$ and $x_j^{(l)}$ are value of the *l*-th dimensional feature in x_i and x_j, and θ_l $(0 \le \theta_l \le 1)$ is used to measure relevance of feature *l*. Obviously, feature *l* is unrelated with this problem when $\theta_l = 0$.

Consequently, the main focus of the algorithm is how to estimate θ_l. Usually, search strategies to estimate θ_l can be classified into three categories: (1) Optimal search; (2) Heuristic search; (3) Random search.

3.2 GA Based Search Strategies

GA is a category of randomized optimization procedures inspired by the biological mechanism of reproduction, which is utilized to search the optimal feature scaling that is most favorable to **SVM** classifier aiming at the feature selection and the improving of performance in our method. The feature scaling is automatically tuned by the maximization of fitness in GA. This is different from other search strategies of feature selection using **SVM** based criteria in the above Refs.. Ref. [10] need Laplacian prior to promote sparse in scaling. Refs. [9, 11] need to calculate the gradient of the criterion (or subject function) with regards to a scaling factor θ_l. However when the subject function is complex, the gradient is difficult to estimate, and the gradient descent algorithm can not always find the global optimal solution. Some weaknesses of above methods can be avoided because of needing no priors in the proposed GA.

A candidate solution to the problem is properly encoded as a string of symbols (e.g., binary) in GA. A set of such string structure the initial population randomly. There are three basic genetic operators used by GA to guide its search inside population: Selection, crossover, and mutation. Selection filters out solutions that perform poorly and choose high performance solutions to concentrate on or exploit probabilistically. Crossover and mutation generate new solutions for exploration. Using the feedback from the fitness function to evaluate and select better solutions, GA converging to a population of high-performance solutions eventually. Although GA does not guarantee a global optimum solution, it has the ability to search through very large search spaces and come to nearly optimal solutions fast. For more detail, refer to Srinivas, *et al.* [12]

3.3 Proposed GA-SVM Method for Estimate θ_l in SVM Classifier

3.3.1 Initial Parameters

A simple encoding scheme is employed, in which the value of a bit binary string is determined by the value of scaling parameter θ_l. Each linked string (or called as citizen) thus represents a different subset of parameter $(\theta_1, \ldots, \theta_n)$.

In general, the initial population is generated randomly. In all of the experiments, we use a population size of 40 citizens and produce 100 generations. In most cases, the GA converges in less than 100 generations.

Selection strategy is cross generational in our methods. Assuming a population of size *N*, the offspring double the size of the population and we select the best *N* citizens from the combined parent–off spring population.

One-point crossover is used, in which the parent citizens are split at a common point chosen randomly and the resulting sub-citizens are swapped.

Traditional mutation operator is used which just flips a specific bit with a very low probability. The mutation probability is set as 0.04.

3.3.2 Fitness Function

The goal of our algorithm is to achieve better performance by estimating scaling parameters in kernels. Therefore, the fitness evaluation contains three terms: (a) accuracy, (b) margin distance, (c) number of eliminated features (where scaling is zero), all of which correspond to a particular subset of $(\theta_1, \cdots, \theta_n)$. We use the fitness function to combine the three terms:

$$Fitness = \alpha_1 \times Accuracy + \alpha_2 \times \frac{numberof\ eliminated\ feature}{numberof\ original\ feature} + \alpha_3 \times Margin \quad (9)$$

Among three terms, accuracy is our major concern. It is estimated by a cross-validation data set which guides the GA search. Accuracy term ranges roughly from 0.00 to 1.00. The second term estimates the proportion of eliminated features, with assuming values ranging from 0.00 to 1.00. Margin distance suggests the general ability of classifier. Its value is problem dependent.

Based on the weights $\alpha_1, \alpha_2, \alpha_3$ that we have assigned to each term, we can select the term to dominate the fitness value mainly. If the accuracy term is decided to control the fitness value more, this implies that individuals with higher accuracy will outweigh individuals with lower accuracy, no matter how many features they contain. This allows driving toward higher classification accuracy than fewer features. In addition, the margin distance in the classification process is used to improve the general ability and provide additional guidance for the GA.

3.3.3 Zero-Bias Operator

In order to predigest complexity of classifier and reduce dimension of feature vector, zero-bias operator is induced to make θ_l near zero more frequently when searching. A binary string is set to 0 randomly with low probability of 0.06. This probability can be adjusted according to problem condition at hand.

3.3.4 Overall Procedure

The overall algorithm, where $P(t)$ is the population of strings at generation t, n is the number of citizen in population, is given below:

Generation $t = 0$
Initialize $\theta_l\ (l = 1, \cdots, n), (0 \le \theta_l \le 1)$, form $P(t)$
Evaluate strings in $P(t)$
    While (termination condition is not satisfied) do
    Begin
    (a) Selection, crossover, mutation and zero-bias operator from $P(t)$ according to fitness

(b) Recombine $P(t+1)$

(c) Evaluate $P(t+1)$ according to fitness

(d) $t = t+1$

End

Output optimal $\theta_l \ (l=1,\cdots,n)$

4 Experiment and Analyze

To evaluate the performance of proposed algorithm to detect spam, experiments are conducted on Spam database in UCI [13] .Standard **SVM** classifier and GA-SVM classifier are used respectively.

4.1 Experiment Datasets

Spam database distinguishes whether an e-mail is spam or nonspam according to 57 attributes (57 continuous, 1 nominal class label). These attributes include percentage of words in the e-mail that match specified word (such as hp, free), percentage of words in the e-mail that match specified character (such as $), average length of uninterrupted sequences of capital letters etc.. There are 4601 instances, in which 1813 are spam. We select 2500 normal e-mails and 1601 spams to construct training set randomly. The remainder 288 normal e-mails and 212 spams are used as testing set.

4.2 Experiment Results

In the experiment, we use standard **SVM** method to train optimal **SVM** classifiers firstly. Under the same kernel type, GA-SVM is utilized to finish feature selection and experiment is performed on feature subset obtained. The performance of these SVMs is measured by the precision of classification, the number of features and support vectors used finally.

The parameters in GA are set as: selection proportion is 0.5, P_c=0.4, P_m=0.04, P_z=0.06. Choosing the weights $\alpha_1, \alpha_2, \alpha_3$ in Eq.(9) for the fitness function is more subjective for user. In the scenario when the best performance is preferred to model cost, the weight α_1 associated with the accuracy term should be very high. Under other different situations when compactness of model is more important than other terms, a higher weight α_2 for the second term should be chosen. In our experiments, values of $\alpha_1, \alpha_2, \alpha_3$ are selected as follow:

$$\alpha_1 = 1, \ \alpha_2 = 0.5, \alpha_3 = 0.5$$

In the spam database, there are five features to be eliminated, such as word_freq_3d, word_freq_addresses, word_freq_857,word_freq_415 and word_freq_table. Tab. 1 gives the results of **SVM** classifier and GA-SVM, in which Poly indicates polynomial kernel function.

Table 1. Result on UCI spam database

Database	Algorithm	Number of Feature	Number of SV	Precision (%)	
				Training	Testing
spam	SVM (Poly2,C=2.3229)	57	919	94.38	87.73
	GA-SVM (Poly2,C=2.5754)	52	905	94.43	87.89

As shown in Tab. 1, comparing with original **SVM**s, the number of features and support vectors decreased ,while better precision are achieved in spam database using GA-SVM. In experiment, we have noted that the training time of GA-SVM is longer than that of original **SVM** because of the search cost of GA.

5 Conclusion

A method to design **SVM** classifier and feature selection jointly using GA for spam detection is proposed in this paper. The relevant coefficients of various features to the classification task, measured by scaling factor, are estimated by GA. And GA approach exploits heavy-bias priors to promote sparse in the scaling factor of features, so feature selection is performed by eliminating irrelevant features whose scaling factor is zero. A **SVM** classifier that has enhanced generalization ability can be learned simultaneously.

Experimental comparisons using original **SVM** and GA-SVM demonstrate that GA-SVM can successfully achieve both of its objectives: better classification accuracy and automatic feature selection. It is also demonstrated that GA can provide a simple, general and powerful framework for tuning parameters in optimal problem, which directly improves detection performance and rate of **SVM**.

References

1. W. W. Cohen.Learning rules that classify e-mail. In Proc. 1996 AAAI Spring Symp. Inform. Access.
2. M. Sahami, S. Dumais, D. Heckerman, E. Horvitz. A Bayesian approach to filtering junk e-mail. In AAAI'98 Wkshp. Learning for Text Categorization, Madison, WI, July 27, 1998
3. Harris Drucker et al. Support Vector Machines for Spam Categorization. IEEE TRANSACTIONS ON NEURAL NETWORKS. 1999,10(5):1048-1054
4. Corinna Cortes, Vladimir Vapnik. Support -vector networks. Machine Learning.1995 (20):273-297.
5. Ron Kohavi, George H. John. Wrappers for feature subset selection. Artificial Intelligence. 97(1997) , 273-324.
6. Isabelle Guyon, Andre Elissee. An introduction to variable and feature selection. Journal of Machine Learning Research . 2003 (3):1157-1182

7. J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, V. Vapnik. Feature selection for support vector machines. Neural Information Processing Systems, Cambridge, MA, MIT Press, 2001
8. Olivier Chapelle, Vladimir Vapnik, Olivier Bousquet , Sayan Mukherjee. Choosing multiple parameters for support vector machines. Machine Learning. 2002 (46):131–159,
9. Alain Rakotomamonjy. Variable selection using SVM-based criteria. Journal of Machine Learning Research. 2003 (3):1357-1370
10. Balaji Krishnapuram, Alexander J. Hartemink, Lawrence Carin, Mario A.T. Figueiredo. A bayesian approach to joint feature selection and classifier design. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2004 (26) 9:1105-1111.
11. Yves Grandvalet, Stephane Canu. Adaptive scaling for feature selection in SVMs. NeuralInformation Processing Systems 15, 2002.
12. M. Srinivas, L. Patnaik. Genetic algorithms: a survey. IEEE Comput, 1994 (27) 6:17–26.
13. Blake, C. L., Merz, C. J. (1998). UCI repository of machine learning databases, URL: http://www.ics.uci.edu/~mlearn/MLRepository.html .

Adaptive Component Allocation in ScudWare Middleware for Ubiquitous Computing

Qing Wu and Zhaohui Wu

College of Computer Science, Zhejiang University,
Hangzhou, Zhejiang, China, 310027
{wwwsin, wzh}@cs.zju.edu.cn

Abstract. With the increasing prevalence of ubiquitous computing, the software component allocation while meeting various resources constraints and component interdependence is crucial, which poses many kinds of challenges. This paper mainly presents an adaptive component allocation algorithm in ScudWare middleware for ubiquitous computing, which uses dynamic programming and forward checking methods. We have applied this algorithm to a mobile music space program and made many experiments to test its performance. The contribution of our work is twofold. First, our algorithm considers resources constraints requirement, component interdependence, and component tolerant issues. Second, we put forward a component interdependence graph to describe interdependent relationships between components. As a result, the evaluation of component allocations has showed our method is applicable and scalable.

1 Introduction

In recent years, computations are becoming ubiquitous and embedded[1], which provide more facilities for our life. Ubiquitous computing aims at fusing physical world and information space naturally and seamlessly, which demands plenty of computation resources for performance requirements. It must require considering interdependences of functional aspects. However, the computation resources in ubiquitous and embedded environments are limited such as CPU computation capabilities, network bandwidth, and memory size. As a result, it sometime cannot provide enough resources to execute some applications successfully. In addition, changes of the heterogeneous contexts including people, computing devices and environments are ubiquitous. Therefore, it results in many problems in software design and development. We think adaptation is the key issue of software systems and applications to meet the different computing environments.

In this work, we focus on the design-time adaptation of component allocation considering resources constraints, component interdependence, and tolerant issues. This paper presents a scalable algorithm using dynamic programming and forward checking methods to allocate components meeting above three aspects in design-time. We have implemented this method for ScudWare middleware, applied it to a mobile music space program, and made many experiments to evaluate its performance. The result shows the method is applicable and scalable.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 1155–1164, 2005.
© IFIP International Federation for Information Processing 2005

The rest of the paper is organized as follows. Section 2 presents a ScudWare middleware platform. Section 3 details a computation model consisting of the adaptive component formalization and a component interdependence graph. Section 4 proposes an adaptive component allocation algorithm. In section 5 and 6, we give a case study and make experiments to evaluate performance and scalability of the algorithm. Then some related work is stated in section 7. Finally, we draw a conclusion in section 8.

2 ScudWare Middleware Platform

We have developed a ScudWare[2] middleware platform in terms of CCM (CORBA Component Model)[3] specification, using ACE (Adaptive Communication Environment)[4] and TAO (The ACE ORB)[5]. TAO is a real-time ORB (Object Request Broker) developed by Washington University.

The ScudWare architecture consists of three parts: ACE components, real-time ORB core and ScudCCM, defined as SCDW=(ACE, ETAO, SCUDCCM). (1) ACE denotes an adaptive communication environment and is targeted for developers of high performance and real-time communication services. (2) ETAO, the extended ACE ORB, is a CORBA middleware framework that allows clients to invoke operations on distributed objects without concern for object location, programming language, OS platform, and communication protocols. ETAO includes a set of services such as transaction, persistence, and life cycle services. In addition, we have developed context-aware, adaptive notification, and component management services added to ETAO. (3) SCUDCCM=(CC, CH, ACM), where CC denotes a set of component containers, which are environments of component runtime; CH denotes a set of component homes, managing component lifetime; ACM denotes adaptive component management. It includes component package, assembly, deploy, and allocation in design-time. Besides, it consists of component migration, replacement, updating, and variation in runtime. Overall, in this architecture, we emphasize on context-aware characters and adaptive mechanisms by providing context-aware service, adaptive service, and component management.

3 Computation Model

In this section, we define an adaptive component computation model in a structural method and describe the problems while meeting resource constraints and component interdependence during the component allocation. We firstly give some definitions about the adaptive component. Then we introduce a component interdependence graph.

3.1 Adaptive Component Formalization

In turn, we give definitions of adaptive component model, adaptive component organization model, adaptive component container model, and adaptive component community model.

Definition 1. *An adaptive component model* $A_C = (C, I, O, R, P)$ is a meta-object. (1) C denotes component's capabilities, including a set of computation functions. (2) I denotes a set of input interfaces provided by other components. (3) O denotes a set of interfaces exporting for other components use. (4) R denotes a set of required resources consumptions value (e.g. CPU computation, network communication bandwidth, and memory size). (5) P is the executing platform of component(e.g. Windows and Linux). Specially, if two components (e.g.a_c^1 and a_c^2) have the same interfaces and capabilities and run at the same platform, we call them component brothers, which is defined as $brother(a_c^1) = a_c^2$. Besides, if two components (e.g. a_c^1 and a_c^2) are not brothers, but having same capabilities regardless of other aspects (e.g. platform feature, interfaces), we call them component friends defined as $friend(a_c^1) = a_c^2$.

Definition 2. *An adaptive component organization model* $A_o = (G, IL, OL, F, RC)$ is a weighed directed component graph. (1) $G \subseteq A_c$ denotes a group of components. (2) IL denotes a set of directed links from the output interface of one component to the input interface of another component. (3) OL denotes a set of directed links from the input interface of one component to the output interface of another component. (4) F specifies the semantic description of IL and OL. (5) RC denotes a set of resource consumption functions as:

(a) *CPU Computation Consumption*: $RC_{cc} : \forall c \in A_c \cdot \exists v \in Q^+ \cdot (RC_{cc} \rightarrow v)$ defines the CPU computation resource consumption by component c. Q^+ is a set of non-negative real numbers.

(b) *Communication Consumption*: $RC_{cm} : \forall c \in A_c \cdot \exists v \in Q^+ \cdot (\sum RC_{cm} \rightarrow v)$ defines communication resource consumption by component c.

(c) *Memory Consumption*: $RC_{mm} : \forall c \in A_c \cdot \exists v \in Q^+ \cdot (RC_{mm} \rightarrow v)$ defines memory resource consumption by component c.

According to above definitions, we have modelled the component resources consumptions as a directed graph. The node weight denotes the CPU computation and memory consumption. In addition, the node links' weight denotes component communications' consumption.

Definition 3. *An adaptive component container model* $A_n = (P, RP, RA, CA, D)$ is a weighed undirected graph. (1) P denotes a set of execution platforms including different OSs and middleware infrastructures. (2) $RP = (RP_{cc}, RP_{cm}, RP_{mm})$ is a set of current left available resources consisting of CPU computation, network bandwidth, and memory resources. (3) $RA = (RA_{cc}, RA_{cm}, RA_{mm})$ is a set of resources that have been allocated for CA. (4) $CA \subseteq A_c$ is a set of components allocated into P. The definitions of RP and RA are same as RC in definition 2. (5) D denotes a set of containers' lifetimes.

Definition 4. *An adaptive component community model* $A_m = (A_o, A_n, K, CT)$ is a component network permeated in ubiquitous computing environments. (1) A_o denotes a set of adaptive component organizations including many heterogeneous components. (2) A_n denotes a set of various adaptive component con-

tainers. (3) K is a set of human-centric tasks. (4) CT is a set of current context information related to application domains.

In terms of above four definitions, we can conclude the component allocation problem formally equals to a model-building problem. That means how to build *an adaptive component community model* A_m by giving *an adaptive component organization model* A_o and *an adaptive component container model* A_n. Specifically stated, there are three restrictions:

(a) Each component is allowed to be allocated in just one container during its lifetime.

(b) It must meet restrictions of resources quantity. That means RP should be more than RA in A_n.

(c) During component allocation, we should consider component interdependence. For example, if component c_1 *has been allocated into container* n_1 and component c_1 and c_2 are interdependent directly, c_2 must be allocated in container n_1 regardless of other aspects.

3.2 Component Interdependence Graph

In order to describe the interdependent relationships between components, we introduce a component interdependence graph composed of component nodes and link paths.

For each component, we associate a node. The link paths are labelled with a weight. We define a component interdependence graph $A_{ig} = (CN, LP, W)$. (1) $CN = \{cn_i\}_{i=1..n}$ denotes a set of component nodes. (2) $LP = \{l_{i,j}\}_{i=1..n,j=1..m}$ denotes a set of component links, describing the dependent targets. $l_{i,j}$ is the link between the nodes cn_i and cn_j. (3) $W = \{w_{i,j}\}_{i=1..n,j=1..m}$ denotes a set of interdependent weight. w_{ij} is a non-negative real number, which labels $l_{i,j}$. In addition, $w_{i,j}$ reflects the importance of the interdependence between the associated components. These weights used, for instance, to detect which links becomes too heavy or if the systems rely too much on some components. In terms of this weight, we can decide which component should be allocated preferentially. Extremely, this graph changes in terms of the different contexts. Therefore, this interdependence is not static: it can be modified when a new component is added or one component disappears. Moreover, based on the different application domain context and run-time environment, the interdependent relationships will change.

4 Adaptive Component Allocation Algorithm

This section describes an adaptive component allocation algorithm, considering resources constraints requirement, component interdependence, and component tolerant issues, which we call *RIT-Based component allocation algorithm*. Besides, this algorithm uses *DP* (*dynamic programming*) and *FC* (*forward checking*) methods, which invokes functions *CCAP*, *SCB* and *ACDC*.

4.1 RIT-Based Component Allocation Algorithm

Giving a set of application domains and human-centric tasks, *RIT-Based algorithm*, listed in algorithm 4.1, aims at generating *an adaptive component community model* A_m from *an adaptive component organization model* A_o, *an adaptive component container model* A_n, and a set of *human-centric tasks* K in terms of resources constraints requirement, component interdependence, and component tolerant issues. First, *RIT-Based algorithm* decomposes tasks into a set of logic functions. Second, it finds a set of key components and their *brothers* and *friends*. Third, it forms the *component interdependence graph*. Fourthly, for each component, it calculates its *allocation priority* according to the resources quantity required and the interdependent relationships between other components. Then it will sort components in terms of their allocation priorities. Specially stated, for each key component, its one brother or friend will be selected as a component backup for tolerant issues. Finally, it completes all component allocations and returns the allocation scheme.

Algorithm 4.1 *RIT-Based Component Allocation*

Input: (1) *An adaptive component organization model* A_o *=(G, IL, OL, F, RC)*; (2) *An adaptive component container model* A_n*=(P, RP, RA, CA, D)*; (3) *A set of human-centric tasks* K*=*$\{lf_1, lf_2, ..., lf_n\}$.

Output: *An adaptive component community model* $A_m = (A_o, A_n, K, CT)$.

Begin

Step 1: *Decomposing* K *into a set of logic functions* $\{lf_1, lf_2, ..., lf_n\}$;

Step 2: *Finding a set of key components and their brothers and friends in terms of* lf_i;

Step 3: *Forming the component interdependence graph*;

Step 4: (1) CCAP : *Calculating all components' allocation priorities and sorting them*; (2) SCB : *Selecting component backups for some key components*;

Step 5: (1) ACDC: *Allocating all components into different component containers, and generating* A_m; (2) *If* ($A_m.A_o.G = A_m.A_n.CA$) *then return* A_m *else return error.*

End.

4.2 CCAP and SCB

For *CCAP*, we introduce an adaptive competence function for each component. We consider both combined resources consumptions of each component and its interdependence. First, for each component $c \in A_o.G$, the component allocation priority *CCAP(c)* is calculated as formula (1).

$$(\alpha * (\frac{RC_{cc}(c)}{\sum_{i=1}^{n} RP_{cc}(A_n(i))}) + \beta * (\frac{RC_{mm}(c)}{\sum_{i=1}^{n} RP_{mm}(A_n(i))}) + \gamma * (\frac{\sum RC_{cm}(c)}{\sum_{i=1}^{n} RP_{cm}(A_n(i))})) * (1 + \sum w(c)) \quad (1)$$

We use α, β and γ to define the weights of CPU Computation, communication and memory resources. If we emphasize CPU Computation more than other aspects, we can set α more than β and γ. In addition, $(1+\sum$ w(c) considers the aspect of component interdependence.

In order to add tolerant mechanism, we introduce the component backup method for key components. In *SCB* step, we select one component brother or friend as its backup. For example, if c_1 is one of key components and has brother c_2, c_2 will be as its backup. If c_1 has not any brothers, but has friend c_3, c_3 will act as its backup. Emphatically, the key component and its backup must be allocated into the different component containers at the same time. For instance, if there are key component c_1 and its backup c_2, and c_1 is placed into component container cn_1, c_2 should be allocated into another component container except cn_1.

4.3 ACDC

For the allocation of each component, we use *DP*(*dynamic programming*) and *FC*(*forward checking*) methods. In *ACDC*, we present three allocation policies:

(a) *If component cn_1 and cn_2 are directly interdependent, we must allocate them into the same component container.*

(b) *Assume that there are two component containers: cn_1 and cn_2. Component c_1 will be placed. If c_1 is to be placed into cn_1, $x_1 = \sum(cn_1.RA/cn_1.RP)$. Also, if c_1 is to be placed into cn_2, $x_2 = \sum(cn_2.RA/cn_2.RP)$. As a result, if $x_1 \leq x_2$, we will allocate c_1 into cn_1, else c_1 will be placed into cn_2.*

(c) *If the allocation of component c_1 dose not meet the resource constraint requirement according to (1) and (2), c_1 will be placed into the component container that has the maximum available resources.*

According to above three policies, we use *DP* and *FC* methods to decide where to place each component, which reduce component allocation complexities.

5 Case Study

In order to demonstrate the usefulness and effectiveness of our component allocation method, we apply algorithm 4.1 to a MMS (mobile music space) program[6] in smart vehicle space. MMS program aims at acquiring music source, playing music, and outputting music adaptively in terms of diverse context information.

The structural model of MMS consists of components for *MSM* (*music source management*), *PM* (*playing music*), *OM* (*outputting music*), *CM* (*context management*), *CA* (*context acquisition*), *CR* (*context reasoning*), *DF* (*detecting fault*) and *CC* (*Controlling center*). Figure 1 shows this structural and resources consumptions of each component and their interdependent weight (Kilobytes for memory and links). There are two component containers: a_n^1 and a_n^2, which are connected via a shared link. Their available resources include 20 and 25 computation, 30 KB and 35 KB memory, and 100 KB/s communication link. As following, we will illustrate the steps of component allocations.

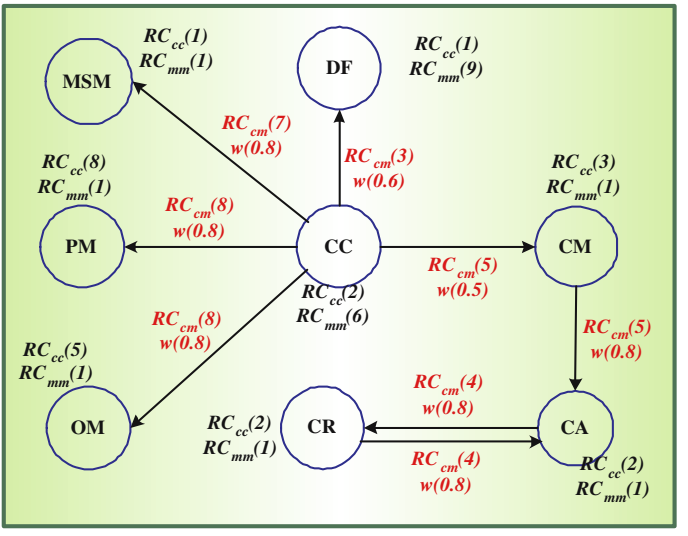


Fig. 1. MMS Component Resources Consumptions and Interdependence Graph

First, we calculate allocation priorities of all components and sort components by formula (1). Specially stated, we assign $\alpha = \beta = \gamma = 1$. Based on the result of calculation, the order of component allocated is (*CC, CM, PM, OM, CA, CR, MSM, DF*), where CCAP(*CC*)=2.0102, CCAP (*PM*)=0.2732, CCAP (*CM*)=0.2377, CCAP (*OM*)=0.2065, CCAP (*DF*)=0.1907, CCAP (*CA*)=0.1796, CCAP (*CR*)=0.1796, and CCAP (*MSM*)=0.1076.

Next, we allocate above components into two containers according to three policies of *ACDC*. Because CCAP(*CC*) is maximum, *CC* is the key component. We select its one brother or friend as a backup to allocate to another container at the same time. In addition, because *CA* and *CR* are interdependent directly,

Table 1. Component Allocation Steps

Steps	Components	X(a_n^1)and X(a_n^2)	Allocation
1	CC(2.0102)	0.3000/0.2514	a_n^2 (CC-backup→ a_n^1)
2	PM(0.2732)	0.5657/0.3823	a_n^2
3	CM(0.2377)	0.2583/0.2357	a_n^2
4	OM(0.2065)	0.3995/0.5596	a_n^1
5	DF(0.1907)	0.1490/0.1370	a_n^2
6	CA(0.1796)	0.2424/0.2607	a_n^1
7	CR(0.1796)		a_n^1 (ACDC: policy (1))
8	MSM(0.1076)	0.1735/0.1496	a_n^2

they must be placed into the same component container. Table 1 shows the component allocation steps and related values. In this case, forward checking is not performed because the allocation of each step meets resources constraints. Emphatically stated, if component c is calculated to placed into a_n^1, however a_n^1 has not enough resources, forward checking is done and component c will be allocated to a_n^2.

Finally, component *CC, PM, CM, DF*, and *MSM* are placed into component container a_n^2, and component *CC-backup, OM, CA,* and *CR* are allocated into component container a_n^1.

6 Experiments Evaluation

In order to test the performance and scalability of above adaptive component allocation algorithm, we have made many experiments. The evaluation uses the number of visited components as a metric for measuring algorithm efficiency. We consider varying the number of both components and containers. For comparison, we choose *SDP* (*standard dynamic programming*) as a baseline, compared to our method *DPFC* (*dynamic programming with forward checking*). The experiments conclude our algorithm has good performance and scalability.

Test 1: Varying the number of components. We evaluate the scalability of the algorithm varying the number of components. First, we fix the number of component containers to be 6, which have different number of resources. Second, we select the number of components from 100 to 1000 in increments of 100, which resource consumptions are randomly generated. Third, we use *SDP* and *DPFC* algorithms respectively to count the number of visited component once a solution found. The result is shown in figure 2. The numbers of visited components caused by *SDP* and *DPFC* both increase as the number of components

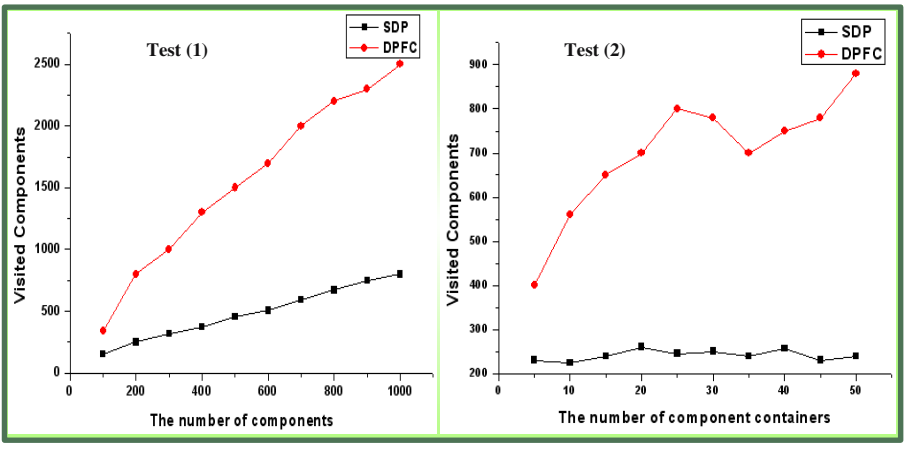


Fig. 2. The numbers of visited components in Test (1) and Test (2)

increases. However, if the number of components is same, the numbers of visited component caused by *SDP* and *DPFC* are different significantly, which is due to forward checking method that improves the solution search.

Test 2: Varying the number of component containers. We evaluate the scalability of the algorithm varying the number of component containers. First, we fix the number of components be 100, which resources consumptions are randomly generated. Second, we select the number of component containers from 5 to 50 in increments of 5, which have different number of resources. Third, we use *SDP* and *DPFC* algorithms respectively to count the number of visited component once a solution found. Figure 2 also shows experiment results. The numbers of visited components caused by *SDP* and *DPFC* both increase as the number of components increase. Nevertheless, on the same number of components, the numbers of visited component caused by *SDP* and *DPFC* are different significantly, which is also due to forward checking method.

7 Related Work

Software component adaptation has the potential for enhancing the system's flexibility and reliability to a very wide range of factors. Adaptive component allocation is playing an important part in software adaptation. Many efforts have been put in this research area.

Shige Wang and Kang G. Shin[7] give a method of component allocation using an informed branch-and-bound and forward checking mechanism subject to a combination of resource constraints. However, their method ignores the execution dependencies of components and tolerant issues when the structural model is partitioned. Belaramani and Cho Li Wang[8] propose one dynamic component composition approach for achieving functionality adaptation and demonstrate its feasibility via the facet model. Nevertheless, they do not consider related aspects in design-time fully. Philip K. Mckinley[9] considers the compositional adaptation enables software to modify its structure and behavior dynamically in response to changes in its execution environment and gives a review of current technology compares how, when, and where re-composition occurs. Kurt Wallnau and Judith Stafford[10] discuss and illustrate the fundamental affinity between software architecture and component technology. They mainly outline criteria for the component integration. Besides, we have proposed a semantic and adaptive middleware[11] and a component management framework for data management in smart vehicle space. According to us, we should consider resources constraints, component interdependence, and tolerant issues for adaptive component allocation in design-time adequately.

8 Conclusions and Future Work

Adaptive component allocation with many aspects' constraints is playing an important role for software component design in ubiquitous computing environments. This paper presents an adaptive component allocation algorithm applied

to the ScudWare Middleware, considering resources constraints, component interdependence, and tolerant issues. This algorithm uses dynamic programming and forward checking methods. We have made many experiments to test the performance and scalability of the algorithm. The experiment results show our method has good performance and scalability.

Our future work aims at improving the performance of our component allocation algorithms. In addition, we will take other methods to realize more flexibility and reliability of component allocation in design-time.

Acknowledgments

This research was supported by 863 National High Technology Program under Grant No. 2003AA1Z2080, 2003AA1Z2140 and 2002AA1Z2308.

References

1. Weiser M: The Computer for the 21st Century. Scientific American, pp. 94-100 (1991)
2. Zhaohui Wu, Qing Wu, Jie Sun, Zhigang Gao, Bin Wu, and Mingde Zhao: ScudWare: A Context-aware and Lightweight Middleware for Smart Vehicle Space. In proceeding of the 1st International Conference on Embedded Software and System, LNCS 3605 pp. 266-273 (2004)
3. http://www.omg.org/technology/documents/formal/components.htm. (2005)
4. http://www.cs.wustl.edu/ schmidt/ACE.html. (2005)
5. http://www.cs.wustl.edu/ schmidt/TAO.html. (2005)
6. Qing Wu and Zhaohui Wu: Semantic and Virtual Agents in Adaptive Middleware Architecture for Smart Vehicle Space. In proceeding of the 4th International Central and Eastern European Conference on Multi-Agent Systems, LNAI 3690, pp. 543-546 (2005)
7. Shige Wang, Jeffrey R. Merrick and Kang G. Shin: Component Allocation with Multiple Resource Constraints for Large Embedded Real-time Software Design. In proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium, (2004)
8. Nalini Moti Belaramani, Cho-Li Wang and Francis C.M. Lau: Dynamic Component Composition for Functionality Adaptation in Pervasive Environments. In proceedings of the Ninth IEEE Workshop on Future Trends of Distributed Computing Systems, (2003)
9. Philip K. McKinley, Seyed Masoud Sadjadi, Eric P. Kasten and Betty H.C. Cheng: Comosing Adaptive Software. IEEE Computer Society, pp. 56-64, (2004)
10. Kurt Wallnau, Judith Stafford, Scott Hissam and Mark Klein: On the Relationship of Software Architecture to Software Component Technology. In proceedings of the 6th International Workshop on Component-Oriented Programming, (2001)
11. Qing Wu, Zhaohui Wu, Bin Wu, and Zhou Jiang: Semantic and Adaptive Middleware for Data management in Smart Vehicle Space. In proceeding of the 5th Advances in Web-Age Information Management, pp. 107-116 (2004)

Prottoy: A Middleware for Sentient Environment

Fahim Kawsar, Kaori Fujinami, and Tatsuo Nakajima

Department of Computer Science,
Waseda University, Tokyo, Japan
{fahim, fujinami, tatsuo}@dcl.info.waseda.ac.jp

Abstract. Our approach towards context awareness is to retrieve contextual information by augmenting our daily life objects (like a chair, a mirror etc.) with sensing capabilities. We call such artefacts *sentient artefacts*. To avoid developing dedicated context-aware application integrating these artefacts, there is a need for a generic computing platform that can assist application programmers to develop and deploy applications easily and rapidly. We present a framework titled "Prottoy"for context-aware applications. The framework provides a generic interface for interacting with sentient artefacts in a unified way, regardless of their type and properties. As a result, application development is simple, rapid and independent from the underlying environments. This paper describes the design and implementation of Prottoy.

1 Introduction

Ubiquitous computing envisions a future environment that will be aware of its operating context and will be adaptive to ease our interaction. Our approach towards such an environment is the environment itself. That means focusing on the building blocks of the environment and making them smart and context aware by capturing peoples implicit interaction. We augment daily life artefacts like a chair, a table, a door, a mirror, a bed etc. with various kinds of sensors to capture contextual information. Our vision is to utilize these objects for value added services in addition to their primary roles. By augmenting sensors, we make these belongings (micro component of the environment) smart. Eventually this process recursively makes our environment smart and context aware in a bottom up approach.

Based on our experiences of developing contextual applications integrating these artefacts, we felt the necessity of a software abstraction that hides the low level details while satisfying requirements like security support, preference reflection, reliability etc. To fulfill these requirements we are working on a software infrastructure called "Prottoy". Our primary goal is to provide a simple, lightweight and extensible framework that offers a unified view of the underlying physical spaces to the applications while supporting all other features.This paper discusses about the design and implementation of Prottoy.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 1165–1176, 2005.
© IFIP International Federation for Information Processing 2005

The rest of the paper is organized as follows: In Section 2, the motivation and design principle of Prottoy are presented. Section 3 gives an insight in the architecture of Prottoy respectively. In section 4 we have presented three sample applications that are built on top of Prottoy. Section 5 discusses several issues and future directions. Section 6 discusses about the related work and finally Section 7 concludes the paper.

2 Motivation and Design Principle

A common goal for programming frameworks for context-aware computing is to provide a computing platform to easily develop and deploy context-aware applications. Programmers can focus on modeling and using the context information while the underlying platform takes care of the actual management and the distribution of this information[1]. There have been several investigations on context aware computing framework in the literature [8] [6] [7] [11] [10] [4] [13] [2] [12] . Many of these frameworks have several interesting features. Prottoys approach is to share all those interesting features while incorporating the features that are missing in other frameworks. In addition Prottoy is designed to satisfy some essential requirements.The major design principle of Prottoy is to provide a generic interface that isolates all environment access issues . Next it targets to separates the application and physical space completely. This separation ensures the decoupling in space and provides fair flexibility to the developers. Another important design issue of Prottoy is to make the application independent of the architecture. Prottoy also designed to be event driven and easily extensible. Final design goal of Prottoy is to provide fair support for context history, end users' preference reflection and artefacts' security.

3 Architecture of Prottoy

As illustrated in Figure 1 Prottoy is composed of few core components and few pluggable components. Core components are the architectural base that facilitate the underlying support for context acquisition and service actuation. Pluggable components run in the application space to assist the developers to exploit the context information.

- Core Framework Components:
  1. Resource Manager 2. Artefact Wrapper 3. Virtual Artefact
- Components Pluggable to Application:
  1. Interpreter 2. Preference Manager

These components internal structures and their corresponding programming models are discussed in the following.

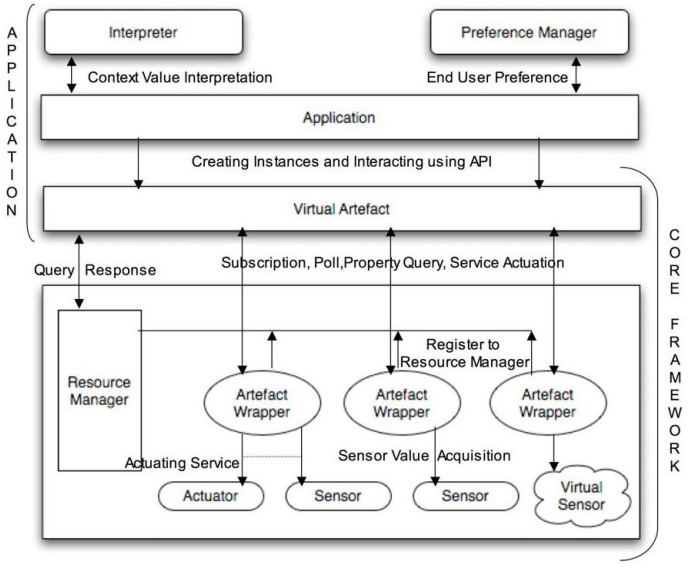


Fig. 1. Basic Architecture of Prottoy

3.1 Resource Manager

It acts as the information repository and service discoverer. It contains artefacts' information like what context or service they provide and their properties. Artefacts register and deregister themselves to the resource manager when they join or leave the environment respectively. Applications locate an artefact by querying the resource manager. If an application uses an artefact, it notifies the resource manager. Also when an application releases an artefact, the resource manager is notified. So the resource manager has the snap shot of the environment and the applications association with the environment all the time.

3.2 Artefact Wrapper

This component encapsulates the sentient artefacts, sensors, actuators and virtual sensors (like weather services, scheduler etc.). It can be used for acquiring context information from the environment and also can be used to actuate services to affect the context or environment. Artefact wrapper is essentially a template that contains various functionalities as already implemented. Artefact developers only need to provide their device drivers for context acquisition or for service actuation. Prottoy handles everything else like event handling, communication mechanisms, deployment techniques etc. Artefact wrapper is composed of the following modules:

1. Communication Module: Artefact wrapper supports two modes of communication. In *synchronous* mode, the context information is constantly dispatched to the interested application whenever the context information is

requested, where as in *asynchronous* mode, the context information is dispatched to the interested applications only when context is changed. In case of the service actuation the artefact wrapper always sends an acknowledgement to the application stating the actuation status.
2. Local Resource Manager: Artefact wrapper has its own resource manager. If the global resource manager in the environment is absent, then the artefact can start its own resource manager to advertise its service or context information.
3. Security Module: Artefact wrapper has an internal security module that allows an artefact to protect its service and information from malicious applications. The artefact only responds to the requests coming from the applications running on its known domain.
4. Device Driver and Context Logic: This module provides a template for the artefact developers to plug in their device driver code and context calculation logic.

Programming Model: Artefact wrapper has built-in features for most of the architectural support. Developers need to provide their device driver and context calculation logic by overriding two functionalities, i.e context update and service execution method. A snippet of artefact wrapper code looks like the following:

```
1. public class Thermometer extends ArtefactWrapper{
2. public Thermometer()
3. public void update(){
4. /* Device driver code and context calculation logic */
5. setContextState(contextStatement);
6. notifyClient();
7. }
8. /* Provide the service execution code here */
9. public synchronized Hashtable executeService(String argument){
10. return null;
11.}
12.public static void main(String[] args){
13. Thermometer artefact = new Thermometer();
14. artefact.start();
15. }}
```

In the `update()` (line 3-7) function, after context calculation, the developer should specify the current context value by setting the context statement using `setContextStatement()` (line 5) and should fire `notifyClient()` (line 6), which eventually notifies all the subscribed applications. The `ArtefactWrapper.start()` (line 14) in the main function initiates the startup process of the artefact.

3.3 Virtual Artefact

Virtual artefact is the heart of our framework that encapsulates the smart environments and provides a unified view to the application developer. Using the

APIs provided in the virtual artefact, the application developers can interact with the actual physical artefacts. Virtual artefact consists of the following components:

1. Communication Module: This module plays the key role for isolating all the access issues to the environment from application point of view. The developer only provides the required artefact functionality/properties using the APIs. This communication model handles the rest by hiding all the details like discovering and establishing the communication link with the actual artefact.
2. Storage Module: Virtual artefact internally hosts a transparent storage component. When an application uses an artefact, the interaction history of the application with the artefact is stored in this storage. Application can later query any historical context.
3. Proxy Module: This module utilizes the storage to facilitate the proxy support when the actual artefact is absent. It produces a calculated context value using some predefined logic. The value is tagged with a low accuracy value indicating that it is a calculated value.

Table 1. Virtual Artefact APIs Available to the Developers

Virtual Artefact API	Functionality
`public VirtualArtefact(String context, String service, PropertyList props, boolean storage, boolean proxy)`	Constructor for creating the virtual artefact instances
`public void subscribe(Object source, String callback)`	For subscribing to artefact for the context information
`public void unsubscribe()`	For removing subscription from the artefact
`public synchronized Hashtable execute (String args)`	For executing artefacts service
`public Hashtable poll()`	For polling an artefacts context information
`public String getProperty(String propname)`	For querying artefacts specific property
`public PropertyList getPropertyList()`	For querying all the properties of an artefact
`Public Hashtable getTimeValue(Calendar time)`	For querying the historical context information

Programming Model: Virtual artefact has methods for all the standard operations required by the applications for interacting with the artefacts. Table 1 summarizes the APIs and their functionalities.

To demonstrate these APIs usage, a very simple application is provided here. The application automatically adjusts a cooler to the comfort level based on the

sensed air temperature.The first 4 lines of the snippet are used to create virtual artefact instances. Then using these instances the actual artefact can be polled (line 6) or can be subscribed to artefact events (line 7). In case of subscription the call back supplied to the framework, is generic and independent of architecture. Also artefact service can be executed (line 17) and properties can be queried (line 9,10) using the instance respectively.

It is visible in the code snippet that applications do not consider network or message management of the architecture and do not need to override or extend any component of the architecture; even applications do not have to look for the resource manager. The developers only provide the context to action mapping rules. Thus Prottoy makes application development fairly simple and rapid.

```
1. PropertyList props = new PropertyList();
2. props.add("location","lambdax");
3. VirtualArtefact thermometer =
             new VirtualArtefact("temperature",null,prop,false,false);
4. VirtualArtefact cooler =
           new VirtualArtefact(null, "cooler service",prop,false,false);
5. If(thermometer.status){
6. System.out.println(thermometer.poll());
7. thermometer.subscribe(this, "thermometerListener");}
8. If(cooler.status){
9. String owner=cooler.getProperty("owner");
10. props=thermometer.getPropertyList();}
11. thermometer.unsubscribe();
12. /*call back*/
13. public void thermometerListener(Hashtable data){
14. /* Hashtable contains context value, timestamp, accuracy and source*/
15. String context = data.get("context").toString();
16. If(cooler.status){
17. System.out.println(cooler.execute("turn-on");}}
```

3.4 Interpreter

This component maps the context value to the interpreted value. We argue that context interpretation is completely application dependent. For example, consider a chair that provides information about its state of use. We can use this information to infer its user is sitting/not sitting (activity) or its users location (at chairs location) based on the applications requirement. Similarly an RFID tag can be interpreted as a person name or a news category name based on the application. Our argument is that, we cannot broadly confine the interpretation of the context information. So we separate it form the core and provide it as a pluggable component at the application layer. The interpreter provides very simple interfaces for the developers to map the context value received from the lower layers to application specific value. During initial deployment , the devel-

oper needs to provide the mapping information, which is later used when the real time context value is received.

3.5 Preference Manager

This component is dynamically generated when the application is deployed and runs at application space. This component is non-editable and is primarily created targeting the end users of the application developed using Prottoy. So we can consider it as a macro running on top of Prottoy. The preference manager provides endusers with the facility to enable or disable the participation of any artefact in the application.

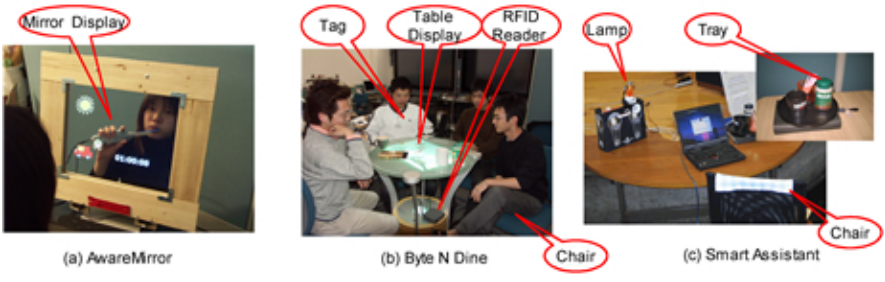


Fig. 2. Sample Applications Developed using Prottoy

4 Sample Applications

While introducing a new framework perhaps the most important issue is the evaluation. We approach this issue by deploying several context aware applications on top of Prottoy. We will discuss here three applications as shown in figure 2 that capture three scenarios at three distinct places namely washroom, dining space and work place.

4.1 AwareMirror

AwareMirror [9] is a smart mirror installed in the washroom. In addition to its primary task of reflecting someones image, it can also provide some useful information related to the person who is using the mirror. It uses two sentient artefacts: a mirror and a toothbrush. It also uses three web services to collect information about users schedule, transportation information and weather forecasting.These services were wrapped in artefact wrapper. When a toothbrush is used, its user is identified and the useful information related to the user is shown on the mirror. Initial goal of this application is to identify artefact wrappers performance on handling heterogonous context sources like physical artefact and virtual artefact (web services). We have found that the artefact wrapper could

handle these successfully. Another important observation was applications effect on artefacts replacement. For testing purpose, weather forecasting web service was replaced by a thermometer and was being simulated to provide the same information as the web service. Interestingly we did not modify single line of code for the proper behavior of AwareMirror. This finding proves Prottoys capability to handle such replacement seamlessly.

4.2 Byte N Dine

This application is designed for a public/private dining space scenario. The goal of the application is to provide the latest news to the user while dining. The application uses two types of sentient artefacts: a few chairs and a table. When the table and the chairs are used in conjunction, the table displays some news. We have assumed that the user will carry a RFID tag that represents his/her preferred topic. The artefact wrapper was used to wrap the artefacts, information source and the RFID reader. Artefact wrapper was successful to handle isolated context source like RFID reader.Another major observation of this application was the development time for such simple applications. Considering reusability of the sentient artefacts, we realized that virual artefact enables writing simple application in a very short span of time.

4.3 SmartAssistant

This application runs in user workspace and can track users activities by analyzing the states of the artefacts populated in the workspace. The application uses a chair, a desk lamp, a tray and a few mugs and jars. Each of these are wrapped in artefact wrapper. The application suggests the user to take a refreshment considering users activity and can provide the user with some predefined schedule notification. Further more, the application can control workspace lighting. This application is more complex in terms of the functionalities than the other two. However we have found that by using virtual artefact interfaces, we could put aside the concern of interacting with such heterogeneous artefacts.One important observation is: context history plays a key role in the analysis of user activity.So such applications are good consumer of historical context.

5 Discussions

In this section we discussed about several issues related to the evaluation of Prottoy and our future work.

5.1 Feedback from Programmers

We have found that application development using Prottoy is fairly simple. None of the applications that we have developed exceed more than 100 lines of code for interfacing with Prottoy. Several members of our lab have used Prottoy in

their applications. Programmers impression on Prottoy is positive. They like its simplicity, especially the virtual artefacts features. They have reported that Prottoy APIs are very easy to use. However they asked for more interfaces at the artefact wrapper layer for the calculation of context.

5.2 Qualitative Evaluation Considering Design Principles

The virtual artefact and the artefact wrapper in conjunction provide the generic interface for everything from a sentient artefact to a single sensor to a web service and to an actuator. The artefact wrapper provides the generalization support that allows the actual artefact to be replaced anytime with another one.While using Prottoy, application developers are free from network management issues. The three-layer architecture separates the application from the physical space completely. The unique callback creation mechanism features of virtual artefact and standard data structure usage provide the complete independence to the application from the architecture. Some of the existing systems provide the storage functionality at the artefact layer [8]. Our argument is that if the artefact itself is absent in that case the storage is also absent. We think the best use of the context storage or the history is the prediction of the context. So it should be somewhere that can be accessible when the artefact is absent. The virtual artefact perfectly solves the problem by hosting the storage. Historical context information stored in the storage can be used for predicting future context or tracking users activity pattern. In Prottoy, the proxy component provides the context prediction support. The proxy service is a unique feature of Prottoy. No architecture yet in the literature supports this feature. Prottoy also provides the support for reflecting end users preference. Current version simply offers end users the flexibility to select an artefacts participation in the application. However our support is very minimal. Another issue is protecting artefact content. We argue that self-descriptive micro world component should be smart enough to protect their content. Some architecture [3] [4] tries to tag the context information as secured or non secured. However malicious applications can still exploit such information. Our approach is different in the sense that the artefacts or the context source can completely deny any request if the requester is not authenticated thus enabling artefact to protect their content from eavesdroppers.

5.3 Major Contribution

Prottoy's event driven design with interesting features facilitate seamless platform for context aware applications. In summary we can point out the following as the main contributions of Prottoy:

1. Generic Interface for all sorts of context source, (namely sentient artefacts, unit sensor, virtual sensor like scheduler, web services etc) and actuators.
2. Complete independence of the application from the underlying architecture.
3. Context storage and proxy service support.
4. Authentication and local resource manager support for the smart artefacts.
5. End users preference reflection support.

Essentially developers need to handle only two components, virtual artefact and artefact wrapper. Because of this, application is very simple and lightweight.

5.4 Future Work

One important drawback of Prottoy is that it does not have any inherent location model for maintaining the spatial relationship among the artefacts. Currently we have used a static location model. However any suitable location model can be adopted in Prottoy with minor modification. We are currently working on this issue. There are few other issues that we are further investigating. Like authentication and proxy module are currently too shallow in performance. We are working on these modules to come up with much better performance The preference component in the current version only provides a selection-based approach. However, we dont feel such GUI based preference is suitable for a context aware application. We are investigating to make this component more realistic and effective. We hope to come up with some interesting results soon on these issues.

6 Related Work

Currently there exist a number of context aware application frameworks in the literature. We have mentioned in Section 2, that Prottoy incorporates many of these frameworks features. However Prottoy also introduces several new notions as we have stated in the paper that differentiates it from the rest of lot. Usually, two approaches have been investigated for context-aware framework. One is the centralized server approach, like Schilits System [12] and the other is the distributed approach like Context Toolkit [8] or Speitzers work [13]. Centralized frameworks provide fair performance from the point of view of context acquisition from the sensors and providing interpreted context via standard APIs. However they suffer from the fact of single point of failure and extensibility concerns. Also, collecting information from several sources in one place makes the framework complex and maintenance becomes difficult. Prottoys approach is different from these as it completely distributes the context sources into multiple artefact wrappers.Thus scalability is well supported in Prottoy.Schilits System [12] deals with the context awareness by Device Agents that maintain the status and the capabilities of the devices, User Agents that maintain the user policies and Active Maps that maintain the location information of the devices and the users. Resource manager and preference component in Prottoy provide the same functionalities. In addition, by introducing features of the artefact wrapper and the virtual artefact, it provides the developer much more control and flexibilities over the physical space and the application development process.

Among the distributed ones, Context Toolkit [8] focuses on the component abstraction by providing the notion of Context Widget and Context Aggregator. Discoverer manages these components and additionally there is a Context Interpreter component that performs the task of context interpretation. Context Toolkit provides very low-level abstraction. Developer needs to provide the

details about the context source like location, port etc. Also, the application is inherently dependent on the framework as the application is tightly coupled with the architecture. That means application needs to extend the architecture component and manipulate accordingly. Prottoy is highly influenced by Context Toolkit, however it differs in several ways. Prottoy takes the Context Toolkit architecture and generalizes it in a single component namely virtual artefact. Using Prottoy, the applications become independent from the context infrastructure. Even the applications do not need to communicate with the resource manager. Applications only use the virtual artefact as a generic component that provides all the infrastructure supports. Prottoy also hides the context implementation from the context specification. It means applications process contextual information at run time dynamically without concerning their acquisition mechanism. In Prottoy, the context storage is at virtual artefact layer that the proxy module utilizes when the actual artefact is absent. We have justified our argument regarding this storage location in the discussion section.

The Stick-e Notes system [5] provides simple semantics for writing rules that specify what action to perform based on the acquired context, mainly focusing on non-programmers to author context aware services. Prottoy generalizes this context acquisition from programmers point of view. The Sentient Computing Project [2] utilizes Active Bat location system to provide an architectural base for indoor application exploiting a world model. Prottoys approach is different from the point of view that it offers the applications to create a context aware environment by constructing an array of artefacts. It means Prottoy specializes the world model creation by allowing developers to construct the model as they want using virtual artefacts.HP Cool Town [7] encapsulates the world by providing web presence of place, people and thing and allows interaction with web presence of these entities primarily exploiting RF technology. Cool Town supports only web based context aware applications where Prottoy supports any classes of context aware applications. Easy Living [6] focuses on an architecture that supports the coherent user experience as users interact with variety of devices in a smart environment. Easy Living also utilizes the notion of world model. Open Agent architecture [11] exploits a centralized black board to support the contextual behavior. In contrast to these systems, Prottoy provides a more generic abstraction as developer has the flexibility to construct the model by manipulating a single component namely virtual artefact.

7 Conclusion

In this paper we have presented a framework "Prottoy" for context-aware application development.When looking at the related work, Prottoy shares many features of the research already done for creating and supporting context information. Our attempt is to take all those frameworks into consideration and come out with one that specializes all. However, Prottoy has several distinct features such as a unique generalized interface, storage and proxy support, security and preference policy. In this paper we have provided the ins and outs of Prottoy

and its approach in a summarized way. We believe Prottoy promises to provide a seamless development platform for context aware application developer.

References

1. G. Abowd. Software Engineering Issues for Ubiquitous Computing. In *21st International conference on Software Engineering*, 1999.
2. M. Addlesee, R. Curwen, S. Hodges, J. Newman, A. W. P. Steggels, and A. Hooper. Implementing a Sentient Computing System. *Cover Feature in IEEE Computer*, 34, 2001.
3. C. Bisdikian, J. Christensen, J. Davis, G. H. Maria R. Ebling, W. Jerome, H. Lei, S. Maes, and D. Sow. Enabling Location Based Applications. In *1st international Workshop on Mobile Commerce*, 2001.
4. J. E. Bradman. Applications of Context Aware Computing in Hospital Work Examples and Design Principles. In *2004 ACM Symposium on Applied Computing*, 2004.
5. J. P. Brown. The stick-e document: A framework for creating context aware applications. In *Electronic Publishing 96*, 1996.
6. B. L. Brumittet, B. Meyers, J. Krumm, A. Kern, and S. Shafer. Easy living: technologies for intelligent environments. In *2nd International Symposium on Handheld and Ubiquitous Computing*, 2000.
7. C. Deborah and P. Debaty. Creating web representations for places. In *2nd International Symposium on Handheld and Ubiquitous Computing*, 2000.
8. A. K. Dey, G. Abowd, and D. Salber. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-aware Applications. *Human Computer Interaction*, 16, 2001.
9. K. Fujinami, F. Kawsar, and T. Nakajima. AwareMirror: A Personalized Display using a Mirror. In *3rd International Conference on Pervasive Computing*, 2005.
10. H. Gallerson, A. Schmidt, and M. Beigl. Adding Some Smartness to Devices and Everyday Things. In *IEEE Workshop on Mobile Computing Systems and Applications*, 2000.
11. C. Philip, A. Cheyer, M. Wang, and S. C. Baeg. An Open Agent Architecture. In *AAAI Spring Symposium Series on Software Agents*, 1994.
12. N. B. Schilit. *System Architecture for Context Aware Mobile Computing*. PhD thesis, Columbia University New York, 1995.
13. M. Spreitzer. Providing Location Information in a Ubiquitous Computing Environment. In *14th ACM symposium on Operating System Principles, ACM Press*, 1993.

Middleware Architecture for Context Knowledge Discovery in Ubiquitous Computing

Kim Anh Ngoc Pham, Young Koo Lee, and Sung Young Lee

Department of Computer Engineering, Kyung Hee University, Korea
anhpnk@oslab.khu.ac.kr, yklee@khu.ac.kr, sylee@oslab.khu.ac.kr

Abstract. Advanced analysis of data for extracting useful knowledge is the next natural step in the world of ubiquitous computing. So far, most of the ubiquitous systems process knowledge in problem-specific or domain-specific manners. This article introduces the concept of context knowledge discovery process, and presents a middleware architecture which eases the task of ubiquitous computing developers, while supporting data mining and machine learning techniques. We show how the middleware architecture supports building ubiquitous systems which are able to "learn" and "think" by introducing some learning – reasoning combination mechanisms, such as the context recognition and prediction, or the deductive rule learning and reasoning process. [1]

1 Introduction

Context-aware ubiquitous computing emphasizes on using context of users, devices, etc. to provide services appropriate to particular person, space, and time. Every computing system dealing with the user should take into account human behavior in one way or the other to materialize ubiquitous computing experiences for him. As we all know that the role of middleware is to ease the task of designing, programming and managing distributed applications by providing a simple, consistent and integrated distributed programming environment; such middleware-based approach is quite appealing in context-aware ubiquitous computing [1].

A lot of work has been done in the area of context-aware computing, in which most of them are only concerned with one or more aspects in an ad hoc manner. Context Toolkit [2] uses the concept of widget to obtain raw contextual information from sensors and passes them either to interpreters or to servers for aggregation. Interpreters and servers use simple HTTP protocol for communication and the XML (name-value pairs only) as the language model for the context. In [3], graph based model for context aggregation and dissemination is proposed where contextual information sources are modeled as event publishers, while context-aware applications as event subscribers. Context Fabric [1] provides a distributed context-aware infrastructure to support the acquisition and retrieval of context data using an entity-relation style logical context data model, encoding data in XML and utilizing

[1] This work was supported by MIC Korea. Dr. S.Y.Lee is the corresponding author.

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 1177 – 1188, 2005.
© IFIP International Federation for Information Processing 2005

XPath as the query language. Gaia [4] is also a distributed middleware infrastructure supporting context-aware agents in smart spaces. It adopted a predicate model of context data encoded in DAML ontologies, and proposed that different logic reasoning and machine learning techniques can be adopted to support context inference.

Based on our knowledge, we have come up with a set of key issues [5]; a context-aware ubiquitous computing system should tackle in order to successfully deploy in real life, namely unified sensing framework, formalized modeling and representation, supporting multiple reasoning approaches, and finally delivering the context to applications on semantic matchmaking basis. And the heart of those intelligent systems is the reasoning and learning capability. Consider a Smart Home scenario, when a user enters his home, the system should be able to recognize who the user is, what he is doing, "guess" what he intends to do, and how he desires the system to assist him. A good reasoning engine can help the system reason properly, but how to build that good reasoning engine turns out to be a difficult question.

Many current Ubiquitous systems are using reasoning engines to infer high-level information from the low-level context data. However, the performance is limited due to the complications in composing rules for the rule-based reasoners, or calculating the uncertainty in probabilistic reasoners. If we can extract some useful rules from the database e.g. "If Bilbo is on bed and it is Bilbo's sleeping time then the probability of Bilbo being sleeping is 85%", then those rules can be used to automatically construct rule-based reasoners. Hence knowledge discovery and data mining come into play. Knowledge discovery and data mining (KDD) deal with the problem of extracting interesting associations, classifiers, clusters, and other patterns from data [10]. KDD is playing an increasingly important role in business, scientific, and engineering applications. In ubiquitous computing environments where large quantity of data is available, advanced analysis of data for extracting useful knowledge becomes the next natural step.

In this paper we propose a middleware architecture for discovering the valuable knowledge from large and heterogeneous amount of context data. We have built and deployed context knowledge discovery architecture as a central part of CAMUS [5] - a middleware infrastructure for context-aware ubiquitous systems.

The rest of this paper is organized as follows: after a brief explanation of context knowledge discovery process in section 2, we present the architecture of CAMUS – our Context-Aware Middleware for Ubiquitous Systems in section 3. In section 4, we give a detailed description of our idea of applying data mining into ubiquitous computing by discussing the rule learning and context recognition mechanism. We conclude our paper with a summary and outlook in section 5.

2 Context Knowledge Discovery

2.1 What is Context Data

Context has been defined by Dey [6] as *any information that can be used to characterize the situation of an entity, where an entity can be a person, place, or a*

physical or computational object which we adopt in this article. A good overview on different definitions of context can be found in [7].

The amount of context data maintained in the system is huge, due to the large number of data sources and from the context history. Since the main task of Data Mining is to extract valuable knowledge from a large amount of data, it is a fundamental requirement for applying data mining in ubiquitous system.

What can we mine from context data? Currently in CAMUS we mine the deductive rules which are used to build rule-based reasoning engines (those reasoning engines then will be used to infer high-level context data from low-level context). We also use clustering technique to discretize the time value and find out the common activities of users. While other systems label the activities in advance, in our system we identify activities by finding clusters of common place, time, environment parameters, and device status. Next section shows the context data mining process, which are somehow different from "traditional" transactional data mining.

2.2 Context Knowledge Discovery Process

Fig. 1 illustrates the context knowledge discovery process and its position in the ubiquitous data inquiry and knowledge management process.

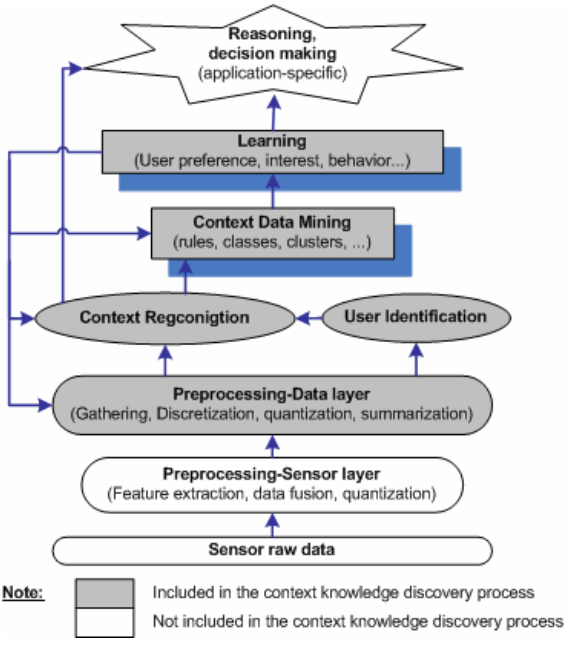


Fig. 1. Context knowledge discovery process and its position in the ubiquitous data inquiry and knowledge management process

Data inquiry and preprocessing at sensor layer: Low-level context data acquired from various sensors is preprocessed by the Feature Extraction Agents (described in

section 3) before sending into Context Repository. This preprocessing step at sensor layer includes feature extraction, data fusion, quantization and encapsulation of sensing data. For example, 3 wireless LAN access points deployed in one area can act as location sensors by extracting and fusing the signal strength. At this step, some features are further quantized into nominal values, for instance the light intensity can have 5 levels of quantized value: dark, dim, normal, bright, very bright.

The extracted features from sensor layer are then forwarded to context data mining process including 4 main steps:

i) Context data preprocessing *(in data layer)*
One important task of this step is gathering relevant context data into data cubes. A context data cube may have multi dimensions: time, location, user, environment parameters (light intensity, sound level, temperature, etc.), device status (television, air conditioner, door status, computer running programs, etc.). The context data has to be user centric; i.e. the environment parameters and device statuses are of the environment where user is currently located in. In order to have a uniform representation of the disparate features, they are further summarized, quantized or discretized, and normalized.

When mining the user preferences or user behavioral habits, user commands to control the devices and appliances are aggregated. Current or scheduled activities are also included into the data cube if available.

ii) User identification and context recognition
Recently, user identification has become an easy task with the advance of identification techniques such as RFID tag, or just simply base on devices associated with the user such as PDA, mobile phones, computer logging on, and the like..

Meanwhile, context recognition remains difficult. Many systems are working on context recognition following different approaches such as using neural network, Bayesian network, etc [17]. Each approach has its pros and cons. In the context data mining process of CAMUS, we use clustering and classification techniques to recognize the context (current activity) of user. The detail of these techniques will be discussed in section 4.

iii) Context data mining
Using data mining techniques to mine context data, this module produces the association rules, classification rule sets and clusters. Output of this module is the input of learning module.

iv) Learning module
This module is an interface between data mining module and the rest of context discovery process. It evaluates the rules and decides which one can be used for reasoning, parses the clusters into classes, maps the rules into the right format (the rule format of the reasoning engine; for example, if we use Jena generic rule reasoners, the rules must have Jena rule format [16]). It also works as a manager to request for new rules, manage the rules and rule mining process, insert or update rule sets of reasoning engines, and so on. The learning module output is used to support all other context data discovery steps: to preprocess data, recognize user context, or

supervise the context data mining. The function of this module will be illustrated by rule learning mechanism in section 4.

Our CAMUS middleware architecture provides maximum support for this context data mining process.

3 CAMUS - Middleware Architecture for Ubiquitous Systems

The knowledge discovery process presented in this paper is the crucial part of our CAMUS architecture, a unified middleware framework for context-aware ubiquitous computing. Here we briefly describe the core functional components of CAMUS as depicted in Fig.2, more details can be found in [5].

3.1 Feature Extraction Agents

Feature Extraction Agents (FX Agents), or wrappers of sensors, extract the most descriptive features for deducing contexts in upper layers, sometimes attached with their semantic meanings and uncertainties. Then, *Feature - Context mapping layer* will perform the mapping required to convert a given feature into elementary context. This module handles the sensor layer pre-processing step in CKDD.

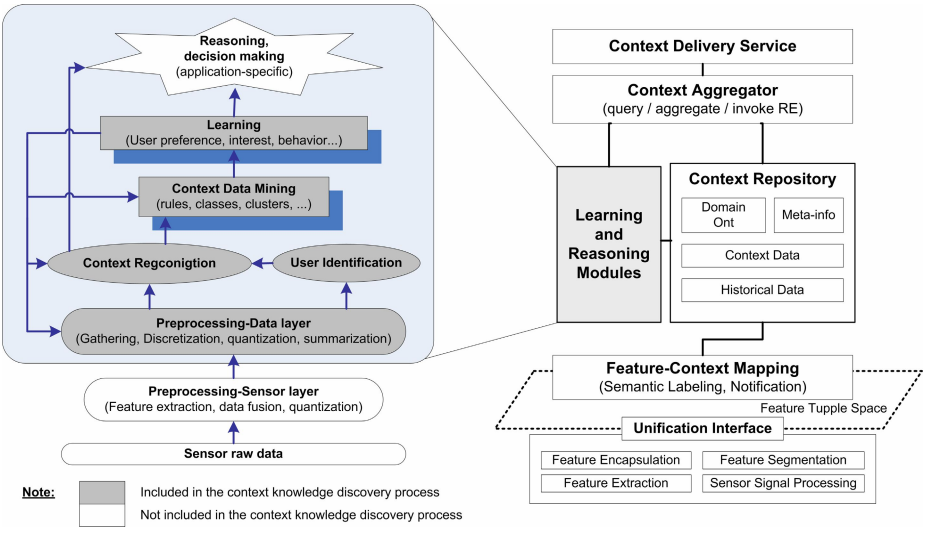


Fig. 2. The correlation between CAMUS architecture and CKDD process

3.2 Knowledge Processing Layer

This module handles the main steps of CKDD process.

Context Repository provides the basic storage services in a scalable and reliable fashion and contains the domain ontology (concepts and properties), contextual

information (including both elementary and composite contexts), and meta-information such as the meta-information about the input, output and capabilities of pluggable reasoning modules.

The Context Repository consists of two databases: the main database which contains all current context data, and the history database. Whenever a new context data comes, the corresponding old context data will be moved into history database. In this way, the size of main database is reduced; consequently it maintains good performance in context query processing.

Reasoning Engine is a collection of various pluggable reasoning modules to handle the facts present in the repository as well as to produce composite contexts. Since not all information can be gathered from sensors, and sometimes the most interesting kinds of context are those that humans do not explicitly provide, context reasoning mechanisms are needed. For instance, the current activity of a user could be inferred based on a combination of many other contexts, e.g. his location, his gestures, time of the day, environment status, etc [5]. Each reasoning mechanism has its own expressiveness, for example Description Logics (DL) is suitable for specifying terminological hierarchies while Spatio-temporal Logic is suitable concerning spatial-temporal sequence in which various events occur, and Bayesian Networks [8] are appropriate for learning the conditional probabilities of different events. Thus a middleware infrastructure needs to provide support for incorporating different reasoning mechanisms into the system, as well as specifying the appropriate mechanism for each context. This will facilitate not only the system internal modules to infer high-level context from low-level or predefined context, but also the applications to reason for their own application-specific context.

To provide more help to developers so that they can concentrate on developing rules or networks for reasoning and not be burdened with the low-level details, our middleware infrastructure defines wrappers for each Reasoner type. For example, a wrapper of Jena generic rule Reasoner allows the developer to easily add a new Reasoner just by declaring the *rule file name* and some *namespace abbreviations*. The following piece of code illustrates how to add and invoke a rule-based reasoner:

```
/* add a new reasoner providing the rule file */
ContextReasonerManager.addReasoner("Location",
ReasonerType.GENERIC_REASONER, "etc/contel.rules");

/* declare some statements */
sms = new ContextStatement[] {PastLocationDescription, hasLocation};

/* invoke the reasoner to do reasoning, providing the reasoner name, the context data name
and the required statements */
cdm.invokeReasoning("Location", "Data", sms);
```

To help constructing the reasoning engine, *Machine Learning Modules* utilize many machine learning and data mining techniques to learn the logic, model (such as user preference model), or train the Bayesian networks and neural networks, using historical context data as training data.

3.3 Context Delivery and Aggregator Services

In our middleware framework, each *context aggregator* (analogous to web service) specifies the context it provides, by utilizing the concepts defined in the ontology repository. This standard schema sharing allows different kinds of entities to be described and utilized by delivery service to find useful services needed by the applications, thus, allowing a flexible mechanism for exchanging descriptive information of various entities.

Context delivery provides the services (context aggregators) with the registration interface to make their information known to the applications. The matchmaking module matches the appropriate service with the client provided the access control policies are not violated.

4 Application of Data Mining Techniques in Ubiquitous System

4.1 Learning the Rules for Context Reasoning

Rule based reasoning is currently the most common form of knowledge processing. Especially in the field of context-awareness, many systems are using semantic knowledge base which fully supports rule-based reasoning engines such as in [4]. However, most developers find building the rules the most difficult task in building ubiquitous computing systems.

In order to minimize the burden for developer in building context-aware applications, our middleware architecture provide support to learn the inference rules from context data and build reasoning engines using those rules, as depicted in Fig. 3.

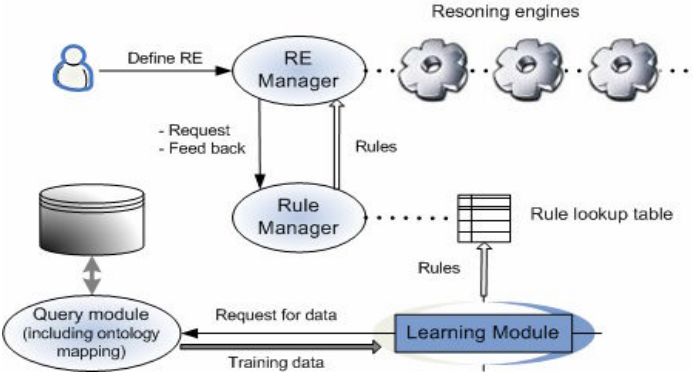


Fig. 3. Rule learning mechanism in CAMUS

To explain the rule learning mechanism in detail, we will first describe a simplified Smart home scenario (only relates to user preference) and how a developer can make use of the rule learning architecture of CAMUS to build a rule-based reasoner which can infer the user preference from context data.

4.1.1 The Simplified Smart Home Scenario

In a Smart Home environment, computing devices are spread throughout the environment, present everywhere in the house to provide services and convenience to the people living inside. The context of user and the status of home environment are sensed by various kinds of sensors, e.g. light sensor, temperature sensor, fire sensor, etc. There are also a set of software agents which control the home devices and provide services to the user seamlessly and invisibly without any explicit user intervention. The following use case is designed focusing on how the Smart Home system adapts to user desire:

Bilbo is sleeping in his bed room an early morning. Since he prefers darkness while sleeping, the system maintains a Dark light intensity level by automatically dimming down the light and rolling down the curtain when the outside becomes brighter. Because normally Bilbo wakes up at around 7:30, at that time the system decides to wake him up and fire the alarm clock.

When Bilbo finishes washing and enters dining room, the television is turned on. Recently he likes to watch the Sport News in the morning, so the system automatically selects the channel which is providing Sport News.

4.1.2 Learning User Preference as Rules

In this scenario, the user preference is learned through user control commands and responses to the messages from system. User can control the home devices by remote controls, or send command messages through some computer software interfaces. The commands are stored in the historical database together with relevant information i.e. user location, timestamp, current activity, environment state. The tuple {user=Bilbo; place=DiningRoom; timestamp = 2005/04/01 8:40; currentActivity=WatchingTV; command= TV.SelectChannel; parameter=SportNews} is an example.

Using this kind of information as the training data, the system can generate rules like:

personName=Bilbo; currenttime=[8:40-8:50]; currentActivity=WatchingTV
⇒ command=TV. SelectChannel; parameter=SportNews (Utility=0.86)

The rules can then be converted into suitable format for the reasoning engine which are used. In our prototype system, because we use Jena generic rule reasoner to reason over OWL[15] format data, the final rule has Jena rule format[16]:

```
[r1:    (?user agt:personName “Bilbo”), (?x time:currentTime ?t),
   (?user act:currentActivity ?curact), (?curact rdf:type act:WatchingTV)
   AND 520 <=?t AND ?t <= 530
   --> [(?pref agt:hasValue “TV.SelectChannel”), (?pref agt:hasPara “SportNews”)
         <-- (makeInstance(?user,agt:hasPreference, agt:Command, ?pref) ]]
```

with agt, time, act, rdf, act are the aliases for the namespaces of ontologies which are currently used to define to data model of our context aware system.

4.1.3 Initiate Rule Learning Process

Because the purpose of rule learning is to supply rule sets for reasoning engines, we first take a look at the information needed to define a rule-based reasoner. Even when the rule set will be learn automatically, the developer must figure out the *set of data* on which the reasoner will do its reasoning. For example, in this smart home scenario, user preference is learn based on user, time, location, environment parameters, user control commands, user activities. Therefore the developer should first define the set of concepts which will be included in the data supplied to the future reasoning engine.

The developer can also define desired *outputs* of the desired reasoning engines. However, the indication of output is optional, because in some cases we don't know exactly the desired output. For example, from activity and other information, we can infer the intended control commands, but also we can base on the user control commands to guess the current activity. In the above scenario, the desired output is user preference.

The Reasoning Engine Manager then sends request to Rule Manager with the required data and/or output of the rules. The learning module will send data request to the query module (part of Data layer Preprocessing module) to get the data needed for mining such rules. Query module responses to learning module with the data cubes.

4.1.4 Rule Mining Algorithm

In ubiquitous systems, a rule mining algorithm has to fulfill following requirements: online (learning has to be done continuously in time domain), adaptive (as user behaviors change over time, rule sets must be adapt to new input data), rules have high accuracy but not necessarily high coverage (For example, every Sunday the user will see a special program on television; the coverage of the control command to select this program is low, but the related rule is highly accurate).

If the output is defined, we use the Sequential-Covering algorithm [9] to learn the first rule sets from example data set. Otherwise, Apriori association rule mining algorithm [10] is more suitable. After learning one rule, the examples covered by the rule will be removed, and then the learning process repeats until no positive example is left.

At this step, a question is raised: How can we select the "right" rules among a large number of learned rules? To address this problem, we assign a utility function to calculate the value of each rule based on confidence and support.

$$Utility(r) = \alpha.Conf(r) + \beta.Sup(r) \tag{1}$$

With *Conf(r)* is the confidence *Sup(r)* is the support of the rule.

The α and β coefficients are related to each other by $\alpha + \beta = 1$, and define the type of rule which is more interested. Normally $\alpha = 1$ and $\beta = 0$, showing that a rule which has high confidence will be chosen even if it rarely happens.

Among a large number of rules mined, only the rules with high utility will be selected. Those rules are inserted into rule table, and supplied to Reasoning Engine Manager to be added into the rule set of the reasoning engine.

Whenever a reasoning engine is invoked, some rules will be fired. The utility of those fired rules can be increased or decreased depending on the feedback from the application after receiving new inferred information is positive or negative. In case of Smart Home system, there are two common types of rule: decision rules and

prediction rules. Decision rule leads to a control command decision, such as turn on the light or adjust the temperature. After the control command is executed, there are two types of response from user:

- If the user satisfies with the automatic command, he will do nothing.
- If the user does not satisfy, he himself will give a control command to adjust the environment as to his desire.

In case of prediction rules, by observing the user, the system can know whether the prediction is right or wrong.

According to the feedback from application, the rules in rule set can be divided into 3 groups: the "right" rules (which generate right prediction or right commands), the "wrong" rules (which generate wrong prediction or unsatisfying commands) and the rules which are not fired.

The utility of those rules then be updated:

$$Utility(r_r) = Utility(r_r) + \gamma \times (1 - Utility(r_r) \tag{2}$$

$$Utility(r_w) = Utility(r_w) - \gamma \times Utility(r_w) \tag{3}$$

$$Utility(r_u) = Utility(r_u) - \varepsilon \times Utility(r_u) \tag{4}$$

Where r_r denote the "right" rule, r_w denotes the "wrong" rule and r_u denotes the unfired rule. γ and ε are the coefficients to tune the speed of changing the utility of rules after each time the reasoning engines are invoked. If ε is set to 0, the utility of unfired rules is maintained.

The rules which have utility below a threshold will be removed from reasoning engine's rule set.

User preference and behavior changes quickly, so the new data plays more important role in the training data set than the old data. To maintain a high learning performance, we can restrict the number of training data set. Only N newest history data records will be retrieved. In that case, whenever new context info comes into the history database and makes the training data set change, the utility of the rules in rule table is updated:

$$Utility(r) = \frac{Utility_{old}(r) \times N + Utility_{new}(r)}{N + 1} \tag{5}$$

Where $Utility_{old}(r)$ is the current utility of the rule and $Utility_{new}(r)$ is the new utility of the rule calculated by the new training data.

Then the examples covered current rules are removed and new rules are mined.

4.2 Context Recognition Using Clustering

In section 2, we briefly described the context recognition step. This section discusses it in detail.

The task of this step is to identify the context (or activity, situation, etc.) of user, based on the low-level contexts from sensors. After preprocessing, the sensor data is gathered in context data cube, including user, time, location, environment parameters and device statuses. To know the context of user, this step handles two tasks:

determining the common contexts of user using clustering over history context data and identifying the current context of user based on sensed data.

4.2.1 Clustering Common User Contexts

Ideally, an algorithm for context recognition must be online and thus unsupervised and must have a variable network topology to cope with changing sensor data cube (changing sensor configurations). The clustering algorithm must not be hard competitive to allow multiple active contexts and it has to be designed for life long learning to not forget or overwrite already learned clusters over time (which is known as the plasticity-stability dilemma [11] in neural networks and clustering literature). Among the most common online clustering algorithms, LLGNG [12] – a modification of GNG algorithm [13] to cope with continuously changing environments and life-long learning - seems to provide most flexibility for context clustering in ubiquitous environment. After the clustering step, the system can label the clusters by itself, or ask user to give some meaningful names, such as "working", "watching TV", "cooking" and the like for the clusters. This is theoretically an optional step, because the machine considers the meaningful and non-meaningful labels the same. However, in practice, while naming the clusters, user can give same name for many clusters, or split a cluster into many different contexts; and hence improve the result of context recognition.

4.2.2 Identify Current Context

This is not the task of data mining but machine learning. Having the result clusters as the context classes, we can apply different machine learning methods to infer the current context/activity/situation of user, base on the time, location, environment and other low-level context data.

Selecting the learning mechanism is up to developer. Our middleware provides the possibility of plugging in different reasoning engines and learning modules. We also provide advance support for the common reasoning engines, such as rule-based reasoning engines (as described above). Another supported approach is Bayesian network, which gains an advantage in the uncertain environment.

CAMUS support Bayesian network by: First, associating each context with a probability attribute; Second, enhancing the context data ontology (implemented in OWL) to represent some concepts which support creating and maintaining Bayesian network, such as relation chain, probabilistic dependencies, etc. [14]; Finally, introducing a Construct-BN algorithm [14] to derive a Bayesian network from the domain ontology.

5 Summary and Outlook

In this paper, we have presented a novel middleware architecture which addresses the key challenges in ubiquitous computing systems, including supports for a context knowledge discovery process. Our CAMUS prototype proves the feasibility of building intelligent ubiquitous environments with lesser workload on the developers.

Our next steps include the effort to make the interface between learning and reasoning more transparent and make more use of data mining techniques.

References

[1] Hong, J. I., et al.: An Infrastructure Approach to Context -Aware Computing. HCI Journal, 2001, Vol. 16.
[2] Context Toolkit project http://www.cs.berkeley.edu/~dey/-context.html
[3] Guanling Chen and David Kotz.: Solar: An Open Platform for Context –Aware Mobile Applications. In Proceedings of the First International Conference on Pervasive Computing (Pervasive 2002), Switzerland, June, 2002.
[4] Anand Ranganathan, Roy H. Campbell: A Middleware for Context -Aware Agents in Ubiquitous Computing Environments. In ACM/IFIP/USENIX International Middleware Conference, Brazil, June, 2003.
[5] Anjum Shehzad et. al. A Comprehensive Middleware Architecture for Context-Aware Ubiquitous Computing Systems. Accepted for publication, ICIS 05.
[6] A. Dey, G. D. Abowd, and D. Salber. A context-based infrastructure for smart environments, 1999
[7] A. Schmidt. Ubiquitous Computing – Computing in Context. PhD thesis, Lancaster University, November 2002.
[8] Korpipaa, P., Koskinen, M., Peltola, J., Makela, S. M., Seppanen, T.: Bayesian approach to sensor-based context awareness. In: Personal and Ubiquitous Computing, Vol. 7, Issue 2. (July 2003) 113-124
[9] Ton Weijters Jan Paredis. Discovering Rules with a Genetic Sequential Covering Algorithm (GeSeCo)
[10] Jiawei Han,Micheline Kamber. Data Mining: Concepts and Techniques
[11] S. Grossberg. Adaptive pattern classification and universal recoding: Parallel development and coding of neural feature detectors. Biological Cybernetics, 23:121–134, 1976.
[12] F. H. Hamker. Life-long learning cell structures continuously learning without catastrophic interference. Neural Networks, 14(4–5):551–573, May 2001.
[13] B. Fritzke. A growing neural gas network learns topologies. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, Advances in Neural Information Processing Systems 7, pages 625–632. MIT Press, Cambridge MA, 1995.
[14] Binh An Truong. Modeling and Reasoning about uncertainty in Context-Aware Systems. Accepted for publication, ICIS 05..
[15] W3C Web Ontology Working Group: The Web Ontology language: OWL. http://www.w3.org/2001/sw/WebOnt/
[16] Jena: A Semantic Web Framework for Java. http://jena.sourceforge.net/
[17] Mozer, M. C. (2005), Lessons from an adaptive house. In D. Cook & R. Das (Eds.), Smart environments: Technologies, protocols, and applications (pp. 273-294). Hoboken, NJ: J. Wiley & Sons

Ubiquitous Computing: Challenges in Flexible Data Aggregation

Eiko Yoneki and Jean Bacon

University of Cambridge Computer Laboratory,
Cambridge CB3 0FD, United Kingdom
{Eiko.Yoneki, Jean.Bacon}@cl.cam.ac.uk

Abstract. A dramatic increase of event monitoring capabilities by wireless sensors requires new, more sophisticated, event correlation over time and space. This new paradigm implies composition of events in ubiquitous computing environments and event Correlation will be a multi-step operation from event sources to final subscribers, combining information collected by wireless devices into higher level information or knowledge. We define generic composite event semantics, which extend traditional event composition with data aggregation in wireless sensor networks (WSNs). This work bridges data aggregation in WSNs with event correlation services over distributed systems. We use interval-based semantics for event detection, defining precisely complex timing constraints.

1 Introduction

In event-based middleware systems, an event correlation service allows consumers to subscribe to patterns of events (composite events). This provides an additional dimension of data management, and improvement of scalability and performance in distributed systems. Particularly in wireless networks, providing event correlation as a middleware service helps to simplify the application logic and reduce its complexity. Event correlation is also important for constructing reactive distributed applications.

The recent evolution of ubiquitous computing has brought with it a significant increase of event monitoring capabilities by wireless devices and sensors. Such systems require new, more sophisticated, event correlation over time and space. The integration of a smart WSN with a large network increases its coverage and potential application domain. In WSNs, a sink node is a sensor node with gateway functions to link to external networks such as the Internet. Sensed information is normally distributed via a sink node. A sink node may also be an gateway node to an intermediate ad hoc network, which may deliver the sensor data to the Internet. Fig.1 depicts WSNs connecting to the Internet, where an ad hoc network conveys sensed data to the Internet.

This new platform enables the seamless use of the various resources in physically interacting environments. A consensus is emerging that the most appropriate system architecture to support such platforms is service management, with communication based on the publish/subscribe paradigm. For example, a publisher broker node can act as a gateway from a WSN, performing data aggregation and distributing filtered data to other networks based on contents. Event broker nodes that offer data aggregation services can

L.T. Yang et al. (Eds.): EUC 2005, LNCS 3824, pp. 1189–1200, 2005.
© IFIP International Federation for Information Processing 2005

coordinate data flow efficiently. Especially when event-based communication is implemented via a peer-to-peer (P2P) overlay network, the construction of event broker grids will extend the seamless messaging capability over scalable heterogeneous network environments. Event Correlation will be a multi-step operation from event sources to the final subscribers, combining information collected by wireless devices into higher level information or knowledge. Mobile devices can be deployed in remote locations without a network infrastructure.

In existing middleware and applications, the semantics of operators for composite events is not defined in a uniform manner leading to a number of problems. Event consumption rules are mostly done as part of an implementation, without a clear semantic definition. Most extant approaches to define event correlation lack a formal mechanism to define complex temporal and spatial relationships among correlated events. Thus, a unified semantics has to be defined to resolve this ambiguity. Temporal ordering in real-time is a critical aspect of event correlation in wireless ad hoc network environments. Neither logical time nor classical physical clock synchronization algorithms may be applicable. In order to determine the direction of movement of a real world entity, temporal ordering of events originating from different devices has to be established. Events can be triggered by physical phenomena, such as glaciers and earthquakes, and the order of occurrence of sensed data is again important.

Event Aggregation, Filtering and Correlation: Some event-based middlewares offer content-based filtering and provide flexible query languages. These allow subscribers to select events of interest, based on the values of their contents. A query can apply to different event types but the aim is to select individual events. On the other hand, event correlation addresses the relationship among, or pattern of, instances of different event types. WSNs have led to new issues to be addressed in event correlation. In WSNs the requirement is to summarize current sensor values in some or all of a sensor network. TinyDB [10] is an inquiry processing system for sensor networks and takes a data centric approach. Each node keeps the data and executes retrieval and aggregation (in-network aggregation), with on-demand based operation to deliver the data to external applications. TinyLIME [3] enhances LIME (Linda In Mobile Environments) to operate on TinyOS. In TinyLIME, LIME is maintained on each sensor node together with a partition of a tuple space. A coordinated tuple space is created across the nodes, connecting with the base station in one hop. It does not currently provide any data aggregation function, only a data filtering function based on Linda/LIME at the base station node. On the other hand, TinyDB supports data aggregation via SQL query, but redundancy/duplication handling is not clear from available documents.

Middleware research for WSNs has been active recently, but most research focuses on in-network operation for specific applications. In this paper, we take a global view of event correlation over entire distributed systems. We define generic correlation semantics, combining traditional event composition and data aggregation in wireless sensor networks. For the event detection semantics, we introduce a parameterized algebra. Parameters include time, selection, consumption, and subset rules. This approach defines unambiguous semantics of event detection and supports resource constrained environments. We introduce interval-based semantics for event detection defining precisely complex timing constraints among correlated event instances. In resource constrained

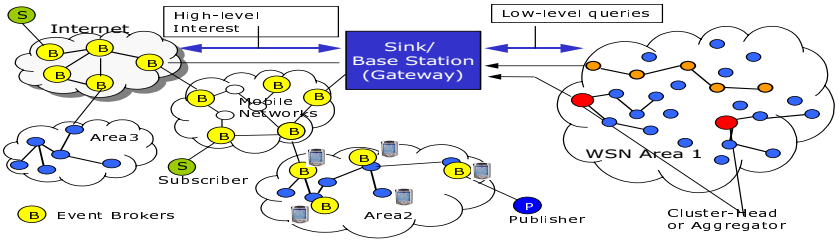


Fig. 1. Bridging WSNs to the Internet

network environments, the event algebra must be restricted so that only a subset of all possible occurrences of complex events will be detected, and this can be achieved by applying appropriate parameters. This paper continues as follows: Section 2 describes the event model, Section 3 defines composite event semantics and section 4 discusses temporal ordering. The formal definition and proof of the event algebra is out of the scope of this paper. Section 5 presents results of experiments. In Section 6 we describe related work, and Section 7 contains conclusions and directions for future research.

2 Event Model

In this section, we define our event model. A primitive event is the occurrence of a state transition at a certain point in time. Each occurrence of an event is called an event instance. Each event has a timestamp associated with the occurrence time. Composite events are defined by composing primitive or composite events with a set of operators.

Timestamps: A timestamp is a mandatory attribute of an event defined within a time system, while the event occurrence time is a real-time defined by the occurrence of the event. Thus, the timestamp is an approximation of the event occurrence time. Most point-based timestamps consist of a single value indicating the occurrence time. In [7], the time when an event is detected is given as an interval-based timestamp, which captures clock uncertainty and network delay with two values: the low and high end of the interval. Although an interval format is used, it represents a single point (point-interval-based timestamp). In addition, we define a (composite) event with duration and give a new interval-based timestamp to a composite event based on interval semantics. A point-interval-based timestamp is an accurate representation, and it is distinct from interval-based timestamps representing the duration of events.

Spacestamp: We introduce spacestamp, as an optional attribute of an event, indicating certain location, relative location, grouping and so forth (e.g., position information (latitude, longitude, elevation) by GPS, global node id). This information can be used for ordering events within the given space.

2.1 Duration

Reference [1] argues that all events have duration and considers intervals to be the basic timing concept. A set of 13 relations between intervals is defined, and rules governing

the composition of such relations controls temporal reasoning. On the other hand, most event systems consider events as instantaneous, that is, the time associated with the event is an instant rather than an interval. A durative event can be seen as capturing the uncertainty over the time of occurrence and the time of detection of an event rather than modelling an event that persists over time. In this sense, durative events are akin to the point-interval-based-timestamps described above. The durative event model considers that instantaneous events are durative events with minimum duration, thus reconciling the models. We would regard an 'event' that persists over time as akin to a state, with an event at the start and one at the end of the time period. This could also be defined as a composite event. Composite events are built up from events occurring at different times, therefore the associated real-time is usually that of the last of its contributory primitive events. This is natural in a context where the prime focus is on event detection, since typically a composite event will be detected at the time that its last contributory event is detected. However, this does lead to logical difficulties in the case of some composite events. Determination of the duration of composite events requires the semantics of composition and time system information such as a point-based or an interval-based time model.

2.2 Duplication

It is important to distinguish between multiple instances of a given event type and duplicates of a given event instance. The expressiveness of some event specification languages has been limited by not distinguishing between event types and instances of those types. [8] attempts to define conditions and constraints on attributes of events in correlation rules rather than defining operators on event instances. Especially in sensor networks, in order to avoid loss of events by communication instability, duplicates of events may be produced to increase reliability. Duplicates have to be handled differently depending on the application, and contexts within applications. For example, in object tracking, the most recent reading from a sensor is valid, and events prior to that will be obsolete, except for the historical record. On the other hand, in a transaction event in which a customer cancels an order, a duplicate event should be ignored because a transaction is being repeated. Thus, the semantics of event composition have to address handling of duplicates. [9] take the approach of defining constraints on attributes of events and detect occurrences of events, before correlation conditions are evaluated. We propose duplicate handling in two ways: adding a selection operator as an event composition operator and adding subset policies as parameters.

3 Event Correlation Semantics

We define composite events by expressions built from primitive and composite events and algebraic operators. The operators of the event algebra are defined informally in this section. We also support parameters, which help to define unambiguous semantics of event detection and support resource constrained environments. In wireless ad hoc networks, the event algebra must be restricted so that only a subset of all possible occurrences of complex events will be detected. We provide basic operators that have the

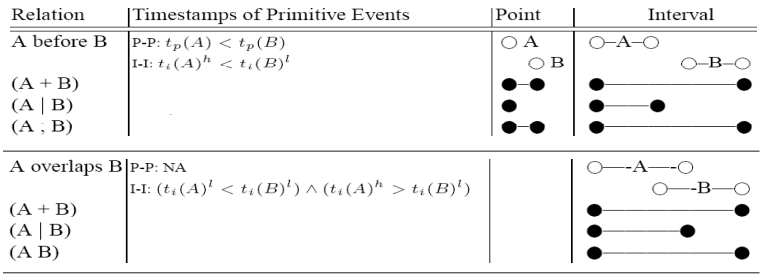


Fig. 2. Temporal Condition for Composite Events

potential of expressing the required semantics and are capable of restricting expressions. Also, an interval semantics supports more sensitive interval relations among events in environments where real-time concerns are more critical, such as wireless networks or multi-media systems. The temporal operators introduced in [1] are not uniformly defined in many applications. We define complex timing constraints among correlated event instances. An example is shown in Fig.2 (see full definition in [16]).

3.1 Composite Event Operators

The event operators are defined informally as follows:

- **Conjunction A + B:** Event A and B occur in any order.
- **Disjunction A | B:** Event A or B occurs.
- **Concatenation A B:** Event A occurs before event B where timestamp constraints are *A meets B, A overlaps B, A finishes B, A includes B, and A starts B*.
- **Sequence A ; B:** Event A occurs before B where timestamp constraints are *A before B, and A meets B*. (A ; B)$_T$ denotes that an interval T between event A and B.
- **Concurrency A||B:** Event A and B occur in parallel.
- **Iteration A*:** Any number of event A occurrences.
- **Negation $-\mathbf{A}_T$:** No event A occurs for an interval T. $(A; B) - C$ denotes that event A is followed by B and there is no C in the duration of (A;B).
- **Selection $\mathbf{A}^N$:** The selection A^N defines the occurrence defined by N. A_T^{AVG} denotes taking the average during an interval T.
- **Spatial Restriction $\mathbf{A}_S$:** Event A occurs if it is a spatial restriction defined in S, that can be defined as a specific location or a group identifier etc.
  - E.g. A_{CB03FD}: The area code *CB03FD* identifies the zone around Computer Laboratory in Cambridge. Event A is valid only when spatial condition is satisfied.
- **Temporal Restriction $\mathbf{A}_T$:** Event A occurs within T. B_T denotes a valid interval for B.

Example: The temperature of rooms with windows facing south is measured every minute and transmitted to a computer placed on the corridor. T denotes a temperature event and T_{30}^{AVG} denotes a composite event of an average of the temperature during 30 minutes. $(T_{room1} + T_{room7})_{30}^{AVG}$ denotes to take an average of room 1 and 7.

3.2 Interval Semantics

We give a timestamp to a composite event based on interval semantics. In most event algebras, each event occurrence, including composite events, is associated with a single time point. This may result in unintended semantics for some operator combinations, for example nested sequence operators. In Fig.3, time flows from left to right, and each row shows the occurrence of a primitive event. When single point detection is used, an instance of event B;(A;C) is detected if A occurs first, followed by B and C. The reason is that these occurrences cause a detection of A;C, which is associated with the occurrence of B;(A;C). With interval semantics, the sequence A;B can be defined to occur only if the intervals of A and B are non-overlapping. No occurrence of B;(A;C) would be detected.

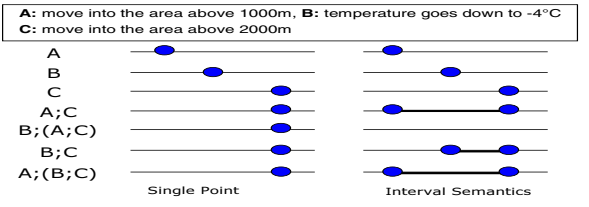


Fig. 3. Point and Interval

3.3 Event Context

Adding the policy defining the constraints provides a way to modify the operator semantics. This parameter-dependent algebra can accommodate different policies on event consumption. First, each operator is given a principle definition of the constraints on the participating occurrences of events that characterize the operator. Then a number of event contexts are defined that act as modifiers to the simple operator semantics. These contexts specify constraints on how occurrences may be selected. As a result, each combination of an operator and a context can be seen as a separate operator with a specific meaning.

Consumption Policy: Three event consumption policies can be defined; *unrestricted*, *recent* and *chronicle*. Snoop [2] uses these contexts, but it is not capable of applying an individual context to different event operators. The parameter dependent algebra clarifies the situations. The following gives an informal definition for detecting A;B.

- Unrestricted: All instances of A and B are valid.
- Recent: If an instance of B can be combined with several instances of A to form instances of A;B, the only recent instance of A is valid.
- Chronicle: Only the oldest instance of A is valid, which is never valid in the future.

Subset Policy: defines the subset of events to detect. Ideally the *Subset Policy* should interfere as little as possible with unrestricted semantics. None of the removed instances should have a crucial impact on the detection of an enclosing detection. At the same time, operations such as conjunction and sequence must be able to identify non-valid

instances early, before the end time of the instance is reached. The main task of the *Subset Policy* is to make an effective algebra, feasible to implement in resource constrained environments. The basis of the *Subset Policy* is that the restricted event stream should be a subset that does not contain multiple instances with the same end time.

Precision Policy: defines the precision of the events to be detected. The dynamic spatial-temporal data from WSNs is generated at a rapid rate and all the generated data may not arrive at the aggregation node over the networks due to the lossy/faulty nature of the sensor network. On the other hand, if some imprecision of the collected data could be tolerated by the application, defining the precision available is important.

For example, *High, Default, Low* can map to:

- the ratio of sensor nodes that are awake: *80%, 20%, 5%*
- the delivered time-series data: *100%, 70%, 50%*
- the interval of data collection: *1 second, 10 seconds, 60 seconds*
- the frequency of data report: *Urgent, Periodical, Available.*

3.4 Event Detection

The current detection mechanism is based on an imperative algorithm, which is executed once every time instant. The main loop selects subexpressions dynamically and computes the current instance of a target composite event from the current event and stored past information. For example, E denotes the event expression to be detected, and subexpressions of E are indexed 1 to k in bottom-up order. The operation result is $E^k(= E)$. Each operation in the expression needs its own indexed state variables (e.g., past events, time instant, and spatial information). A canned detection component can be created for common use. We implemented a prototype using a simple automata with support of parameterized values and time constraints.

4 Temporal Ordering

Sensor networks are used to monitor real world phenomena and for such monitoring applications, physical time is crucial. In global computing environments, such sensor data flow over heterogeneous networks. We cannot assume a global clock, or globally synchronized physical clocks, to correlate events. Moreover when the store-and-forward paradigm is used for communication, message propagation delay is unavoidable. Traditional message ordering based on a transport layer protocol is not applicable. Thus, we use timestamps embedded in events for correlation, which provide a real-time mechanism. Temporal ordering of events is highly influenced by the event detection method, timestamping methods and the underlying time systems.

In many real world scenarios, wireless networks may be deployed with relay nodes to the Internet and it is possible that relay nodes can connect to Global Positioning System (GPS). GPS may be the key for providing accurate time adjustment at certain nodes that are less resource constrained within wireless ad hoc networks. We define two categories of network environments; where NTP is deployed with GPS, such as the Internet domain, and where networks are isolated in ad hoc mode without GPS or any other deterministic time synchronization mechanism. For the first category, we use

interval-based point timestamps for primitive events, where the interval low and high end values are computed as described in [7] to allow for clock uncertainty and network delay. For timestamping composite events we use interval-based semantics, unlike[7] where a new timestamp is taken on the detection of a composite event. For the second category, several time synchronization mechanisms have been proposed. Among those, we investigated the one described in [14]. The idea of the algorithm is not to synchronize the local computer clocks of the devices but instead to generate timestamps from a local clock. When such locally generated timestamps are passed between devices, they are transformed to the local time of the receiving device. We propose a simplified protocol *Lightweight Local Clock Propagation*, which is on-demand based timestamp synchronization. The basic idea is that each node calculates its processing time using a local clock, and at the sink node, the sum of the processing times is subtracted from the event arrival time to estimate the occurrence time. Comparable timestamps are therefore created at sink nodes instead of network-wide. This requires the two assumptions: network delay is negligible (e.g. the node is close to the radio or network deployment is dense) and clock drift is negligible (e.g. the node carries an oscilloscope that guarantees less than 10 ppm drift).

Thus our proposal is a coordinated approach with and without the use of GPS. Sensor events could be aggregated at gateway nodes with transformed timestamps and passed towards a subscriber node in the Internet environment, where GPS-based time synchronization is deployed.

5 Experiments

Below are experiments results for composite event detection with time restriction, subset rule, and an object tracking system. Surface meteorological data from the NOAA Aeronomy Laboratory TRMM profile system recorded in 1999 are used as base data for the experiments described in Section 5.1-2. The data contain wind, temperature, relative humidity, pressure and solar radiation. The data from each instrument were sampled every 0.5 second. Every 10 seconds, a 10 second average was transmitted to a base station. As part of the 10 second average, a timestamp was added to the data. The clock is kept in UT time, and is set to the GPS time standard every week. The logged data were assembled to the individual event and a discrete event generator simulates the event occurrences as if sensed data are reported from the sensor network in single hop communication range. In the first experiment, each event is duplicated. There is no loss of events in both experiments. The composite event used in the experiments is as follows: humidity below 60% (H) followed by temperature over 30% (T) within 10 seconds, which denotes $(H;T)_{10}$.

5.1 Time Restriction

Consider as an example, the event expression $(H;T)_4$: before the current time instant 10, there have been six occurrences of H. If a T_4 instance occurs in the current time instance, and the start time of this instance is 9 or 10, it should be combined with h_6 (occurred at time instant 7) to form an instance of $H;T_4$. If the start time is 8, it should

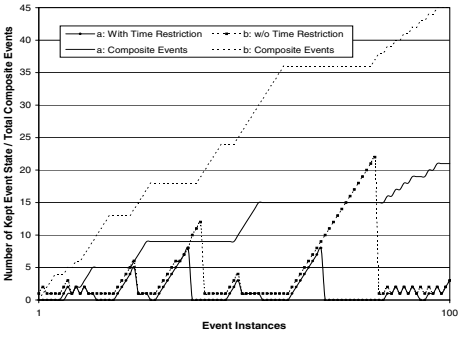


Fig. 4. Time Restriction

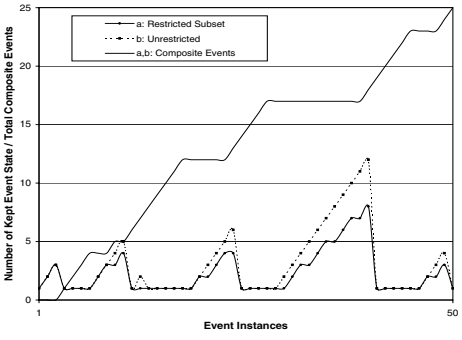


Fig. 5. Subset Rule

be combined with h_5 (at 6), etc. Since an instance of T_4 with an end time of 10 must start no earlier than 6, it follows that it must be combined with either h_3 (at 4), h_4 (at 5), h_5, or h_6, and thus there is no need to store h_1 (at 1) and h_2 (at 2). Throughout the detection of this expression, all instances of H that end more than 4 time units ago, except the one with the latest start time, can be discarded. Fig. 4 shows a simulation of memory usage with the number of event states to be kept. For the detection policy, the most recent instance of H is used. The time restriction value is set to 10, which may be relatively a large number. The time restriction is a similar concept of event detection with a sliding window, but with our approach, semantics are unambiguous and capability of individual definition for each event gives another advantage. Our approach ensures that detection of composite events can be efficiently implemented with limited resources, which can be an critical element for embedded applications. The number of detected composite events is also illustrated in Fig. 4 that shows only half of composite events being detected when time restriction is specified. If undetected composite events imposes loss of information, then the time restriction number should be reduced.

5.2 Subset Rule

Fig. 5 illustrates the memory usage when the H's subset H_S. The event instances with the same end time is removed and the Subset Rule keeps exactly one with maximal start time. Because the composite event $(H_S; T)$ is equivalent to $(H; T)$, in an environment with the Subset Rule all the composite events, supposed to be detected in unrestricted environments should be detected. In Fig. 5, both cases show the same results for composite event detection. On the other hand, with the Subset Rule less states are kept.

5.3 Object Tracking

We developed a prototype of an object tracking system using the Active BAT data (see [17] for more detail). Fig.6 shows the specific period, when two people are positioned at the machine room. The composite event *(Brian+Andy)machine-room*, where 'Brian and Andy are together at the machine room' is detected between the time unit 1523 and 1529.

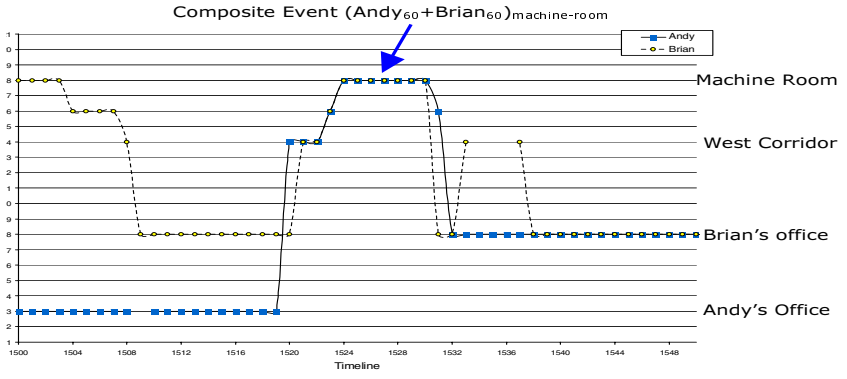


Fig. 6. Composite Event Detection

6 Related Work

Much composite event detection work has been done in active database research. SAMOS [5] uses Petri nets, in which event occurrences are associated with a number of parameter-value pairs. The transition from centralized to distributed systems led to the need to deal with time. [2] presents an event-based model for specifying timing constraints to be monitored. [9] proposes an approach that uses the occurrence time of various event instances for time constraint specification. GEM [11] allows additional conditions, including timing constraints, to combine with event operators for composite event specification. Composite events service in an event-based middleware system reduces the communication within the system and potentially gives a higher overall efficiency, which is addressed in [12]. Hayton et al. [6], on composite events in the Cambridge Event Architecture, describe an object-oriented system with an event algebra that is implemented by nested push-down FSA to handle parameterized events.

Temporal message ordering has been an issue in traditional networks such as for system monitoring and in distributed event systems. In existing systems, the semantics of event order often depends on the application logic. For real-time support, a common solution in wired networks provides a virtual global clock that bounds the value of the sum of precision and granularity within a few milliseconds. The 2g-Precedence model is enhanced for distributed event ordering and composite event detection using 2g-precedence-based sequence and concurrency operators [15]. However, in open distributed environments, not all servers are interconnected and event ordering based on NTP may lead to false event detection. Interval-based time systems define event order based on intervals. In [7], timestamps of events can be related to UTC (Universal Coordinated Time) with bounded accuracy, and event timestamps are modeled using accuracy intervals. They use NTP that provides reference time injected by a GPS time server and, in addition, returns reliable error bounds. For wireless network environments, [13] presents a GPS based virtual global clock, which is used for timestamping events, and deploys a similar concept to 2g-precedence. Post-facto synchronization [4] is based on unsynchronized local clocks but limits synchronization to the transmit range of the mobile nodes.

7 Conclusions and Future Work

Our event correlation semantics supports a new paradigm coming from the recent evolution of ubiquitous computing with a rapid increase of event monitoring capability by wireless devices. The main focus is on supporting time and space related issues such as temporal ordering, duplicate handling, and interval-based semantics, especially for wireless network environments. Event management will be a multi-step operation from event sources to final subscribers, combining information collected by wireless devices into higher-level information or knowledge in a global computing environment. Work is ongoing on the transformation of event algebra, so that complex expressions can be more efficiently implemented in resource constrained devices over wireless ad hoc networks. We are working on a complete implementation, including various timestamping environments.

Acknowledgment. This research is funded by EPSRC (Engineering and Physical Sciences Research Council) under grant GR/557303.

References

1. Allen, J. et al. Maintaining Knowledge about Temporal Intervals. *CACM*, 26(11), 1983.
2. Chakravarthy, S. et al. Snoop: An expressive event specification language for active databases. *Data Knowledge Engineering*, 14(1), 1996.
3. Curino, C. et al. TinyLIME: Bridging Mobile and Sensor Networks through Middleware. *Proc. PerCom*, 2005.
4. Elson, J. et al. Wireless Sensor Networks: A New Regime for Time Synchronization. *Workshop on Hot Topics in Networks (Hotnets-I)*, 2002.
5. Gatziu, S. et al. Detecting Composite Events in Active Database Systems Using Petri Nets. *Proc. RIDE-AIDS*, 1994.
6. Hayton, R. et al. OASIS: An Open architecture for Secure Inter-working Services. *PhD thesis, Univ.of Cambridge*, 1996.
7. Liebig, C. et al. Event Composition in Time-dependent Distributed Systems. *Proc. 4th CoopIS*, 1999.
8. Liu, G. et al. Composite Events for Network Event Correlation. *Proc. IFIP/IEEE International Symposium on Integrated Network Management*, 1999.
9. Liu, G. et al. A Unified Approach for Specifying Timing Constraints and Composite Events in Active Real-Time Database Systems. *Proc. RTAS*, 1998.
10. Madden, S. et al. TAG: A tiny aggregation service for ad-hoc sensor networks. *Proc. of Operating Systems Design and Implementation*, 2002.
11. Mansouri-Samani, M. et al. GEM: A Generalized Event Monitoring Language for Distributed systems. *IEE/IOP/BCS Distributed systems Engineering Journal*, 4(2), 1997.
12. Pietzuch, P. Shand, B. and Bacon, J. Composite Event Detection as a Generic Middleware Extension. *IEEE Network Magazine*, 18(1), 2004.
13. Prakash, R. et al. Causality and the Spatial-Temporal Ordering in Mobile Systems. *Mobile Networks and Applications*, 9(5):507-516, 2004.
14. Roemer, K. Time Synchronization in Ad Hoc Networks. *ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc01)*, 2001.

15. Schwiderski, S. Monitoring the Behavior of Distributed Systems. *PhD thesis, University of Cambridge*, 1996.
16. Yoneki, E. and Bacon, J. Unified Semantics for Event Correlation over Time and Space in Hybrid Network Environments. *Proc. CoopIS*, 2005.
17. Yoneki, E. and Bacon, J. Object Tracking using Durative Events. *Proc. NCUS*, 2005.

Author Index

Abderazek, Ben A. 77, 340
Amamiya, Makoto 538
Amirijoo, Mehdi 131
Angelov, Christo 152
Asakawa, Kazuo 3
Atiquzzaman, M. 1099

Bacon, Jean 1189
Baek, Yunju 663
Bai, Hongjie 291
Baik, Doo-Kwon 858
Bao, Lichun 639
Barolli, Leonard 1123
Bhalla, Subhash 203
Bihler, Pascal 977
Botía, Juan A. 957
Brandt, Jens 405
Brunie, Lionel 977
Bu, Yingyi 920
Bulcão-Neto, Renato F. 900

Chae, Heeseo 987
Chandra, Punit 560
Chang, Jyh-Huei 255
Chang, Pao-Chi 805, 1006
Chang, Tae-Gyu 458
Chaudhary, Vipin 360
Chen, Chien-Chen 37
Chen, Hanhua 1075
Chen, Juan 267
Chen, Jui-Ming 683
Chen, Shaxun 920
Chen, Song 291
Chen, Yuan-Hsiu 489
Chen, Yung-Yu 947
Cheng, Ming-Chun 765
Chi, Hsin-Chou 67
Chiu, Ge-Ming 693
Chiu, KauLin 703
Cho, Jinsung 597
Cho, Jun-Ki 174
Cho, Yeong-Hun 673
Cho, Yongyun 879
Choi, Byeong-Doo 174
Choi, Hae-Wook 300
Choi, Ho-Seung 629
Choi, Jaeyoung 879
Choi, Jun Kyun 1065
Choi, O-Hoon 858
Chong, Kiwon 499
Chou, Hung-Jen 947
Chou, Li-Der 683
Chu, Slo-Li 281
Chung, Kwang-Sik 715
Chung, Sung Woo 331
Conti, Vincenzo 27
Cuong, Doan Huy 1065

Ding, Jen-Wen 651
Dittmann, Florian 448
Dodonov, Evgueni 1133
Doh, Yoonmee 142
Dohi, Tadashi 245
Dong, Sheqin 291
Dong, Yong 267
Doose, David 184
Dou, Wenhua 224
Du, Changrui 87
Du, Junzhao 745
Duda, Andrzej 438
Durresi, Arjan 1123
Durresi, Mimoza 1123
Duval, Sébastien 826

Falaki, Mohammad Hossein 311
Fong, Anthony S. 47, 371
Fujinami, Kaori 1165
Fukuda, Hisashi 1036
Fusayasu, Takahiro 1036

Gao, Jerry 528
Goh, John 735
Gómez-Skarmeta, Antonio F. 957
Götz, Marcelo 469
Gu, Zonghua 214
Guha, Dipnarayan 1065
Guo, Minyi 550

Hallenborg, Kasper 846
Han, Joohyun 879
Han, Sangman 597
Hansson, Jörgen 131
Hasegawa, Masaki 203
Hashizume, Hiromichi 826
Higuchi, Hayato 428
Ho, Shih-Pang 683
Hofmann, Richard 234
Hong, Xianlong 291
Hou, Meijia 1055
Hsieh, Ching-Tang 1107
Hsiung, Pao-Ann 479, 489
Hsu, Chi-Yu 775
Hu, Chia-Shing 1107
Hu, Jun 395
Huang, Min 1055
Huang, Shyh-Fang 805, 1006
Huang, Yueh-Min 651
Hwang, Chong-Sun 715
Hwang, Min-Cheol 174
Hwang, Soyoung 663

Ide, Masanobu 1016
In, Hoh Peter 987

Jan, Rong-Hong 255
Jarnikov, Dmitri 930
Jeong, Dongwon 987
Jeong, Moon-Sang 673
Jha, Niraj K. 1
Jhon, Chu Shik 57
Jiang, Jie 224
Jiang, Peilin 1026
Jin, Hai 1075
Jin, Ying 1045
Jo, Seng Kyoun 1065
Jung, Chulho 836
Jung, SoonYoung 715
Jung, Won-Yong 350

Kang, Chang-Duk 795
Kao, Tzu-Han 947
Kasano, Kazuhiko 1036
Kawata, Sotaro 77, 340
Kawsar, Fahim 1165
Kim, Bongi 1036
Kim, Chang-Soo 867
Kim, Chang-Soon 795
Kim, Cheol Hong 57
Kim, Daeyoung 142
Kim, Dongsun 98
Kim, Hag-Young 867
Kim, Hyun-Gyu 17
Kim, Jae-Hyun 629
Kim, Jaeeon 142
Kim, Jeongmin 321
Kim, Jin-Hyung 174
Kim, Jin-Sam 174
Kim, Jongmyon 4
Kim, Juil 499
Kim, Mi-Ae 940
Kim, Mijeom 967
Kim, Myung-Joon 867
Kim, Sangwook 996
Kim, Shin-Dug 795
Kim, Youngsoo 142
Ko, Sung-Jea 174
Komuro, Hiraku 1016
Kong, Ki-Sik 715, 725
Kshemkalyani, Ajay D. 560, 572
Kumar, Mohan 967
Kung, Hsu-Yang 775, 785
Kuroiwa, Shingo 1026
Kyusaka, Shinya 428

Larkin, Henry 609
Law, K.L. Eddie 890
Lee, Eunseok 509, 836
Lee, Hyunju 996
Lee, Sang Ho 816
Lee, Sanghee 836
Lee, Seongsoo 350
Lee, Seunghwa 509
Lee, Sungyoung 597, 1177
Lee, Won-Hyung 940
Lee, Woojin 499
Lee, Yong-Jin 1099
Lee, Young Koo 1177
Lentini, Giuseppe 27
Leveugle, Régis 1085
Li, Jun 920
Li, Keqiu 518
Li, Xuandong 395
Lim, Jung-Eun 858
Lin, Mei-Hsien 775
Lin, Wei-Hsuan 947
Lin, Yen-Cheng 683
Lin, Yingjun 550
Liu, Chia-Ni 775

Liu, Lei 1045
Liu, Meilin 121, 164
Liu, Zhen 1147
Lu, Jun 639
Lukkien, Johan 930
Lv, Chuanfeng 910
Lv, Jian 920

Ma, Jianhua 1016
Mammeri, Zoubir 184
Marian, Nicolae 152
Mello, Rodrigo Fernandes de 1133
Mine, Tsunenori 538
Mohsen, Amjad 234
Mok, Pak Lun 371
Musfiquzzaman, Akanda Md. 77

Nagamoto, Taichi 418, 428
Nakajima, Tatsuo 110, 1165
Nambiar, Anilkumar 360
Ngo, Vu-Duc 300
Nguyen, Huy-Nam 300
Nieto, Ignacio 957
Nilsson, Daniel 131
Norin, Henrik 131

Oguri, Kiyoshi 418, 428
Oh, Hyeong-Cheol 17
Oh, Jehwan 509
Oikawa, Shuichi 110
Okamura, Hiroyuki 245
Oprescu, Justinian 438

Park, Dong-Jo 194
Park, Jong-Tae 673
Park, KwangJin 715, 725
Park, Kyung-Lang 795
Park, Sooyong 98
Paruchuri, Vamsi 1123
Pereira, Carlos Eduardo 469
Pham, Kim Anh Ngoc 1177
Pimentel, Maria da Graça C. 900
Portolan, Michele 1085
Pu, Dongbing 87

Qiu, Meikang 121, 164
Qu, Wenyu 518

Rahayu, Wenny 755
Ren, Fuji 1026
Rettberg, Achim 448, 469
Rousseau, Franck 438
Ruiz, Pedro M. 957
Ryu, Sang-Moon 194

Sánchez López, Tomás 142
Sarker, Biplab Kumer 584
Schneider, Klaus 405
Scuturici, Vasile-Marian 977
Seo, Gang-Hoon 350
Sha, Edwin H.-M. 2, 121, 164
Sha, Kewei 745
Shao, Zili 121, 164
Shen, Hong 518
Shen, Shih-Hao 651
Shi, Weisong 745
Shibata, Yuichiro 418, 428
Shieh, Wann-Yun 37
Shin, Soong Hyun 57
Shina, Yusuke 1016
Shirazi, Behrooz 967
Shu, Lei 597
Shyu, Shys-Rong 1107
Sierszecki, Krzysztof 152
So, Sunny 890
Song, MoonBae 715, 725
Song, Peisong 1036
Song, Zhen 224
Sorbello, Filippo 27
Sowa, Masahiro 77, 340
Srinivasan, Bala 755
Suda, Tatsuya 639
Sugaya, Midori 110
Suh, Hyo-Joong 321, 331
Sung, Jongwoo 142

Takata, Katsuhiro 1016
Tan, Pengliu 1075
Tan, Yi Yu 47, 371
Tanaka, Masataka 1016
Tanaka, Yoshito 1036
Taniar, David 735, 755
Tao, Xianping 920
Tavakoli, Ahmadreza 311
Teixeira, Cesar A.C. 900
Tešanović, Aleksandra 131
Ting, Magdalene P. 528
Toomey, Warren 609
Tripathi, Anil Kumar 584

Tsai, Tsung-Han 947
Tseng, Chih-Hao 479

Uchida, Mitsuru 418
Uehara, Kuniaki 584

Vahdatpour, Alireza 311
van der Stok, Peter 930
Vidyarthi, Deo Prakash 584
Vitabile, Salvatore 27

Waluyo, Agustinus Borgy 755
Wang, Huai-bin 1147
Wang, Guojun 550
Wang, Junwei 1055
Wang, Tsan-Pin 703
Wang, Xingwei 1055
Weber, Raphael 448
Wee, Jae-Kyung 350
Wei, Jiun-Ju 785
Wills, D. Scott 4
Wills, Linda M. 4
Wu, Bin 572
Wu, Che-I 785
Wu, Chia-Ming 67
Wu, Di 518
Wu, Eric Hsiao-Kuang 805, 1006
Wu, Fu-Lan 693
Wu, Qing 1155
Wu, Xiaoling 597
Wu, Zhaohui 1155
Wu, Zheng da 609

Xiang, Hua 1026
Xu, Li 1075
Xu, Xizhen 458
Xue, Chun 121, 164

Yamane, Satoshi 381
Yang, Jie 597
Yang, Laurence Tianruo 584, 1133
Yang, Xue-jun 267
Yano, Satoshi 418
Yau, Chi Hang 47, 371
Yi, Hui-zhan 267
Yi, Myung-Kyu 619
Yoneki, Eiko 1189
Yoo, Jae-Soo 867
Yoo, Jin Ho 816
Yoo, Kil-Sang 940
Yoo, Seongeun 142
Yoshinaga, Tsutomu 340
Young, Cheng-Ru 693
Yu, Haibo 538
Yu, Wing Shing 47, 371
Yu, Xiaofeng 395
Yu, Ying 1147
Yu, Zhezhou 87
Yuan, Shyan-Ming 765, 947

Zhang, Heying 224
Zhang, Minghu 1075
Zhang, Tian 395
Zhang, Yan 395
Zhao, Qiangfu 910
Zheng, Guoliang 395
Zheng, Xiao-juan 1045
Zhou, Chunguang 87
Zhuge, Qingfeng 121
Ziavras, Sotirios G. 458